February 10, 2015, v3.2

MATLAB BASED LANGUAGE FOR GENERATING RANDOMIZED MULTIPLE CHOICE QUESTIONS

NURULLA AZAMOV

Abstract. In this work we describe a simple MATLAB based language which allows to create randomized multiple choice questions with minimal effort. This language has been successfully tested at Flinders University by the author in a number of mathematics topics including Numerical Analysis, Abstract Algebra and Partial Differential Equations.

Contents

1. Introduction

This manual gives a description of a special purpose programming language which I named Spikě. Spikě is a language specifically designed to create randomized multiple choice tests in mathematics and other subjects. Spikě is written and is based on MATLAB. The output of Spikě is a LaTeX file which contains an assignment. This presumes that a potential user of Spikě is to be familiar to a certain extent with both MATLAB and LaTeX.

Spikě is free but to use it you have to have MATLAB and LaTeX installed in your computer. Since university teachers of mathematics and physics normally have MATLAB and LaTeX, in case they decide to use Spikě it should cost them nothing.

I wrote the first version of this language for my personal use a few years ago. Obviously, before plunging into such a task I browsed the Internet in a hope to find something like this which would be easy to learn and easy to use. But I did not find anything like this and hence I decided to write such a language myself. I successfully used it in three mathematics topics which I teach and I decided that others may find it useful too.

Before proceeding further, I would like to say that my attitude towards testing knowledge by using *only* multiple choice tests is strongly negative, but as part of knowledge testing process multiple choice tests have many advantages among which are cost-effectiveness and absolute unambiguity of the marking process. These qualities make multiple choice tests an almost indispensable tool.

To start with let us consider some simple examples.

1.1. **Examples.**

1.1.1. *Example of* `problemI` *type question.* Assume that we wish to test whether a child has learnt how to add two-digit numbers. Normally, we would randomly choose two two-digit numbers, for instance 45 and 83 and ask the child to add them.

The following Spikě-program does something similar.

```
<<problemI;
  a={10:99};
  b={10:99};
  answer('$%d$',a+b);
@2; a,b;
Find the sum of numbers $%d$ and $%d.$
>>
```

What does this program do? This program picks up randomly two integer numbers a and b from the interval 10..99, for instance, 45 and 83 and prints the text of the question

Find the sum of numbers 45 and 83.

Further, the command `answer('$%d$',a+b);` tells Spikě that the correct answer is $a + b$ and so it calculates the correct answer 128. To create a multiple choice question we have to pick up, say, four incorrect answers. The identifier `problemI` which follows after the opening spike-bracket `<<` tells Spikě how to create these four alternative incorrect answers: it picks up another pair of integers a and b from the interval 10..99 and adds them to get the first alternative incorrect answer, then it repeats this process three more times. Each time Spikě checks whether the alternative incorrect answer is new, that is, different from previous ones, to ensure that all multiple choice answers are different. Finally, Spikě randomly permutes these answers. The result of processing the above code is the following piece of LaTeX text.

```
\problemtype{I}
\problA{D}{2}
Find the sum of numbers $74$ and $48.$

\medskip\noindent
(A) \ $120$ \quad
(B) \ $142$ \quad
(C) \ $143$ \quad
(D) \ $122$ \quad
(E) \ $39$ \quad
```

This is a piece of a LaTeX-file. Here `\problemtype` and `\problA` are LaTeX commands from the preamble of the output LaTeX-file which you may define as you wish. The argument `{D}` of `\problA` indicates that the correct answer is `D`. The second argument in `\probl{D}{2}`, that is, the number 2, is supposed to show a student how many marks this question is worth. This second argument is taken from the at-line.

All is left to do is to convert this LaTeX-file into a PDF-file. The following is a piece of this PDE-file.

1. [2m] Find the sum of numbers 74 and 48.

(A) 120 (B) 142 (C) 143 (D) 122 (E) 39

This output can be easily modified by changing definitions of LaTeX commands `\problemtype` and `\problA`.

1.1.2. *Example of* `problemR` *type question.* Now assume that we wish to test whether a high school student has learnt how to solve quadratic equations. To partially achieve this aim we can use the following code.

```
<<problemR;
  a={2:8};
  b={2:8};
  c={2:8};
  x2=(-b+sqrt(b^2+4*a*c))/(2*a);
  answer('$%.3f$',x2);
@2; a,b,c;
```

```
Find the largest root of the quadratic equation $%d x^2 + %d x - %d = 0.$
Present this root with five digits after the decimal point.
>>
```

The following is a piece of LaTeX-file produced by this Spikě-code.

```
\problemtype{R}
\problA{D}{2}
Find the largest root of the quadratic equation $5 x^2 + 7 x - 5 = 0.$
Present this root with five digits after the decimal point.

\medskip\noindent
(A) \ $0.52060$ \quad
(B) \ $0.52062$ \quad
(C) \ $0.52064$ \quad
(D) \ $0.52066$ \quad
(E) \ $0.52068$ \quad

\problemtype{R}
\problA{A}{2}
Find the largest root of the quadratic equation $2 x^2 + 5 x - 3 = 0.$
Present this root with five digits after the decimal point.

\medskip\noindent
(A) \ $0.50000$ \quad
(B) \ $0.50002$ \quad
(C) \ $0.50004$ \quad
(D) \ $0.50006$ \quad
(E) \ $0.50008$ \quad
```

The following is a piece of processed PDF file.

2. [2m] Find the largest root of the quadratic equation $5x^2+7x-5=0$. Present this root with five digits after the decimal point.

(A) 0.52060 (B) 0.52062 (C) 0.52064 (D) 0.52066 (E) 0.52068

3. [2m] Find the largest root of the quadratic equation $2x^2+5x-3=0$. Present this root with five digits after the decimal point.

(A) 0.50000 (B) 0.50002 (C) 0.50004 (D) 0.50006 (E) 0.50008

The difference between type I and type R problems is in the way Spikě creates alternative incorrect answers. (In the third question above the answers are not good, of course)

2. How does Spikě work?

2.1. **How to obtain Spikě code?** Until 1st of September 2015 you can email me requesting the spike code. Please write in the subject of your email "request for spike code". You will get an automatic reply with spike m-files attached.

Another option is to copy-paste the spike-code given in Appendix B.

2.2. **Installing Spikě .** There is no setup program which could install Spikě in your computer, so it should be installed manually. Fortunately, this is not difficult. In addition, as Jamie Oliver once said, an advantage of cooking your own meal is that you know exactly what ingredients are in it.

To instal Spikě , carry out the following steps:

(1) In MATLAB folder `MATLAB\` create a new folder and name it `Spike\`. We shall refer to this folder as *spike folder*.
(2) Copy files `spike.m` and `topfile.tex` into the spike folder.
(3) Create folder `Spike_m_files\` in the spike folder.
(4) Copy the files
   - `assignment.m`,

- `datablock.m`,
- `problem.m`,
- `problemG.m`,
- `problemH.m`,
- `problemI.m`,
- `problemR.m`,
- `problemT.m`,
- `problemV.m`.
- `spikeblock.m`,
- `topic.m`,
- `ProcessCurvedBrackets.m`,
- `ReadSpikeBlock.m`,
- `SeparateVariables.m`,
- `Spike2Matlab.m`,

into the folder `Spike_m_files\`.

In order to use Spikě the following steps need to be carried out as well:

(1) Create a folder with topic name in the spike folder. This folder will be called *topic folder*.
Normally, university and college topics have topic codes. For example, at Flinders University Algebra has topic code `MATH2712`, Partial Differential Equations has topic `MATH3712`, etc. I usually choose the topic code to name the topic folder.
(2) Create a file with the list of university ID numbers of students enrolled in the topic. The exact format of this file is explained below.
(3) Create folder `Utils\` in the spike-folder. In this folder we shall keep Matlab functions used in Spikě files.

2.3. **To get started.** Spikě is a computer program written in MATLAB, and it is executed by MATLAB's processor. The aim of this program is to produce a LaTeX file which contains an assignment. Once the assignment LaTeX file is ready all that is left to do is to convert it into a PDF file and then post it online for students to view. This also means that to use Spikě it is absolutely necessary to have MATLAB in your computer, but it is not necessary to have LaTeX in it. You need LaTeX if you want to do something with the LaTeX file produced by Spikě.

Any assignment has two attributes: assignment's topic and assignment number. Hence, not surprisingly, Spikě has two arguments: topic code and assignment number. Assume that the topic code is `MATH2712` and the assignment being prepared is the third one. Assuming that all Spikě files have been properly installed, in MATLAB's prompt `>>` you should type the following command:

```
>>spike('MATH2712',3)
```

and then press [Enter] key.

This will work if

(1) The current folder is the spike-folder; this is the folder which contains the m-file `spike.m`.
(2) The current folder has a subfolder `Spike_m_files` which contains the following m-files: `assignment.m`, `datablock.m`, `problem.m`, `problemG.m`, `problemH.m`, `problemI.m`, `problemR.m`, `problemT.m`, `problemV.m`. `spikeblock.m`, `topic.m`, `ProcessCurvedBrackets.m`, `ReadSpikeBlock.m`, `SeparateVariables.m`, `Spike2Matlab.m`.

These files form an integral part of the Spikě-code and without any of them Spikě will not work. A user of Spikě do not need to worry about the content of these files. There is one exception: in the file `topic.m` there is code which defines topic names given their topic codes. One may wish to change this code or otherwise leave it as it is, since it is not essential anyway.
(3) The current folder contains a folder whose name coincides with the topic code, that is, `MATH2712\`. This folder will be referred to as *topic folder*.
Spikě saves the output LaTeX file with the assignment in this folder. The output LaTeX file is automatically given a name in the format `TopicCodeAN.tex`. For example, in our case the output file will be named `MATH2712A3.tex`.
(4) The topic folder contains a file with a list of students' ID's enrolled in the topic. The name of this file should have the format `TopicCode.txt`. For example, in our case there must be a file

`MATH2712.txt` in the topic folder `MATH2712\`. This file will be referred to as *student list file*. The content of this file can be as follows:

```
2006280
2032971
2062720
2084704
-------
```

This means that there are four students enrolled in the topic `MATH2712` and these are their student IDs. As a result, Spikě will prepare four assignments. A file with list of students must contain a line which starts with "seven dashes" `-------`. Everything which follows after this line is ignored by Spikě. This is convenient, since if there are say 150 students enrolled in a topic you would not prepare 150 assignments every time you test your Spikě-code unless everything has been tested and the final version of the assignment can be prepared. This is important since Spikě works extremely slowly: it takes about quarter of a second to prepare one multiple choice question. So, if an assignment contains 20 questions and there are about 50 students, it may take Spikě a few minutes to prepare an assignment.

For example, I keep all student ID's in a student list file and insert the second seven dashes line after the second or the third student ID, so that Spikě prepares only two or three assignments. When everything is ready I remove this seven dashes line.

(5) In the topic folder there is a file which contains the Spikě-code for the assignment. The name of this file has strict format: `TopicCodeAN.spk`. For example, in our case the topic folder must have a file `MATH2712A3.spk`. How to create this kind of `spk`-file is what this manual is about.

A spike-file consists of spike-blocks `<<...>>`, a *spike-block* is what comes between *spike-brackets* `<<` and `>>`.

For example, the following is a piece of an actual spike-code which contains two spike blocks.

```
<<data;
  p={7,11,13};
  g=randi(p-1,1,3);
  sg=PolyToLaTeXDesc(g);
>>

<<problemI;
  f=randi(p-1,1,2);
  h=randi(p-1,1,2);
  sf=PolyToLaTeXDesc(f);
  sh=PolyToLaTeXDesc(h);
  fh=polyprod(f,h,p);
  [~,r]=DART(fh,g,p);
  sr=PolyToLaTeXDesc(r);
  answer('$%s+\\langle %s \\rangle$\\\\',sr,sg);
@2,0; sf,sg,sh,sg,p,sg;
Find the product of elements $%s+\langle %s \rangle$
and $%s+\langle %s \rangle$ of the ring $\mbZ_{%d}[x]/\langle %s \rangle.$
>>
```

(6) The current folder must have a file with name `topfile.tex`. This file will be called *top-file*. The top-file contains assignment information such as (1) due date for the assignment, (2) instructions how to answer assignment questions, (3) definitions of LaTeX commands such as `\problA`, etc. It is up to you what you write in the top-file.

Before the line `\documentclass...` in the top-file it can also have some MATLAB commands initializing certain parameters, such as the maximum mark for the assignment, due date, etc.

For example, the spike folder in my computer contains the following folders and files:

```
MANUAL\
MATH2712\
MATH3701\
MATH3712\
Spike_m_files\
SpikeMark\
```

```
STAT0000\
Utils\
spike.m
topfile.tex
```

One can infer from this that I teach three topics (actually four, but for Topology I don't use Spikě); these topics are Algebra, Numerical Analysis and Partial Differential Equations.

The following is the content of `MATH2712\` folder:

```
Final\
MATH2712A1.spk
MATH2712A2.spk
MATH2712A3.spk
MATH2712A4.spk
MATH2712A1.tex
MATH2712A2.tex
MATH2712A3.tex
MATH2712A4.tex
MATH2712.txt
```

One may wonder about the purpose of the folder `Final`. The thing is that Spikě overwrites an old spike file if there is one. Hence, if one is absolutely happy with the final version of the assignment and the assignment is ready to be posted online for students to view, then it is very desirable if not necessary to copy the final version to a secure place where Spikě does not poke its nose. The folder `Final` is that secure place.

2.4. **How Spikě works?** Spike does the following:

(1) First Spikě sets default values of certain parameters. These parameters can be changed in the top-file.
(2) It reads in the topic-data, such as list of student ID's.
(3) It creates for writing the out-file `TopicCodeAN.tex`.
    In our case it is the file `MATH2712A3.tex`. This file is saved in the topic folder, which is, in our case, `MATH2712`.
(4) Spikě opens the top-file and executes MATLAB commands which precede the line starting with `\documentclass`.
(5) Spikě copies the LaTeX code from the top-file into the out-file.
(6) And finally, Spikě creates the assignment by processing spk-file.

3. Writing Spike-programs

3.1. **Spike-blocks.** In this section we explain how to prepare a spike file. A spike file consists of a sequence of spike blocks. A *spike block* is a text which comes between *spike-brackets* `<<` and `>>`. Spike-brackets should be placed at the beginning of a line. If `<<` or `>>` is found somewhere inside a line, then it is not considered as a spike-bracket.

There are three types of spike-blocks: text-blocks, data-blocks and question-blocks. In their turn question-blocks can be one of the following six types: I, R, T, G, H and V.

There are six types of multiple choice questions which Spikě can prepare: type I, type R, type T, type G, type H and type V. These letters stand for words "integer", "real", "true", "generate" and "verbatim" respectively. Each question is prepared by a question-block of the appropriate type.

A question-block has the format

```
<<problemX;
  ...
>>
```

where `X` is one of the six letters I, R, T, G, H or V. How to prepare a question-block will be explained in subsequent subsections.

Spikě reads a spike file until it encounters the first spike block, processes it and then proceeds reading the file until it finds the second spike block, processes it and so on.

Anything that is placed outside a spike-block is ignored by Spikě. Hence, one can use space between spike-blocks for comments.

3.2. **Text-blocks.** If a spike-block is a text-block, then Spikě copies the content of the block into the output LaTeX file.

A text-block has the format

```
<<text;
  (Some text to be written
   into the output file)
>>
```

For example, if we are about to prepare a spike-block which creates a question about the Central Limit Theorem, then we may write such a text-block:

```
<<text;
  \bigskip
  {\bf The Central Limit Theorem.}
>>
```

One can write LaTeX commands in a text-block too, of course.

3.3. **Data-blocks.** A data-block has the format

```
<<data;
  (Matlab commands defining, say, some variables
   to be used in other spike-blocks)
>>
```

In a data block we can define some variables. For example, assume that a question asks to convert a number from base 10 to base b. We can define the base b in a data-block as follows:

```
<<data;
  b = {2:8};
>>
```

The command `b = {2:8};` assigns to b a random value from the interval 2..8. Then the variable `b` can be used in all subsequent spike-blocks, until it is redefined or removed.

For example,

```
<<problemI;
  x={1000:9999};
  y=Convert_to_base(x,b);
  answer('\\verb!%s!$_{%d}$ \\',y,b);
@2; x,b;
  Convert the number $%d$ to base $%d.$
>>
```

In this type I spike-block `Convert_to_base` is a Matlab function which converts an integer x to base b and returns the result as a string in LaTeX format. These kind of Matlab functions used in spike-blocks I keep in the folder `Utils\`, though they can be kept anywhere else. This is what this spike-code produces in the output LaTeX file:

```
\problemtype{I}
\problA{D}{2}
  Convert the number $9815$ to base $4.$

\medskip\noindent
(A) \ \verb!1321113!$_{4}$ \ \quad
(B) \ \verb!122201!$_{4}$ \ \quad
(C) \ \verb!2012322!$_{4}$ \ \quad
(D) \ \verb!2121113!$_{4}$ \ \quad
(E) \ \verb!113101!$_{4}$ \ \quad
```

This is the corresponding part of the PDF file:

4. [2m] Convert the number 9815 to base 4.

(A) $\texttt{1321113}_4$ (B) $\texttt{122201}_4$ (C) $\texttt{2012322}_4$ (D) $\texttt{2121113}_4$ (E) $\texttt{113101}_4$

This text is a piece of assignment for one of the students. In the next run the value of the variable b may change to another, since the command `b = {2:8};` will be executed anew. So, the next student may get the following question:

```
\problemtype{I}
\problA{A}{2}
  Convert the number $6460$ to base $5.$

\medskip\noindent
(A) \ \verb!201320!$_{5}$ \ \quad
(B) \ \verb!140214!$_{5}$ \ \quad
(C) \ \verb!33402!$_{5}$ \ \quad
(D) \ \verb!114242!$_{5}$ \ \quad
(E) \ \verb!223010!$_{5}$ \ \quad
```

5. [2m] Convert the number 6460 to base 5.

(A) 201320_5 (B) 140214_5 (C) 33402_5 (D) 114242_5 (E) 223010_5

One may ask why not to define the variable b inside the `problemI`-block as follows:

```
<<problemI;
  b = {2:8};
  x={1000:9999};
  y=Convert_to_base(x,b);
  answer('\\verb!%s!$_{%d}$ \\',y,b);
@2; x,b;
  Convert the number $%d$ to base $%d.$
>>
```

This is what we get if we do this:

```
\problemtype{I}
\problA{A}{2}
  Convert the number $9895$ to base $2.$

\medskip\noindent
(A) \ \verb!10011010100111!$_{2}$ \ \quad
(B) \ \verb!12246!$_{7}$ \ \quad
(C) \ \verb!111110211!$_{3}$ \ \quad
(D) \ \verb!32054!$_{6}$ \ \quad
(E) \ \verb!111011121!$_{3}$ \ \quad
```

6. [2m] Convert the number 9895 to base 2.

(A) 10011010100111_2 (B) 12246_7 (C) 111110211_3 (D) 32054_6 (E) 111011121_3

3.4. **Questions of type I and type R.** Questions of types I and R (I for "integer" and R for "real") have the following structure:

```
<<problemI;
  (Command part)
  answer(...);
@N1[,N2[,N3]]; [parameters]; [NAltAns;]
  (Text of question)
>>
```

or

```
<<problemR;
  (Command part)
  answer(...);
@N1[,N2[,N3]]; [parameters]; [NAltAns;]
  (Text of question)
>>
```

There is very little difference between type I and type R questions, so we will call them type IR questions.

Here the (Command part) is the command part of a type IR question, `N1` is a one-digit number which is treated as number of points given for a correct answer to the question, `N2` is a one-digit number which is treated as number of negative points given for an incorrect answer to the question, and `N3` is a one-digit

number which is treated as number of negative points given for an unanswered question, (Text of question) is text of question which may (in fact, should) contain some randomized variables, and (parameters) are parameters of the text part of the question. The parameters `N1` and `N2` are not compulsory.

The command part consists of a sequence of MATLAB commands, but some of those commands can be special Spikě commands which are translated to MATLAB commands by Spikě before feeding them to MATLAB's processor. The aim of these commands is

(1) to randomize certain elements of the multiple choice question,
(2) to find the correct answer to the question,
(3) to generate four alternative incorrect answers.

In current version of Spikě all MATLAB commands should be written in one line. For example, you can use double-nested `for`-cycle but you should fit them into a single line. This is a bit inconvenient but since one can use MATLAB functions in the command part this is not a big problem.

The text part of type IR question is the text of the multiple choice question which is typed in LaTeX format and some elements of which are replaced by values of variables given in the command part. For example, a question asking to add three integer numbers may have the following text part:

```
Find the sum of numbers 47, %d and %d.
```

This means that the first of these three numbers is not randomized while the other two are. The commands `%d` and `%d` are replaced by values of two variables which are given in the second `@`-parameter, for example, `a,b;`. This means that the first number `%d` will be replaced by the value of the variable `a`, and the second number will be replaced by the value of the variable `b`.

A type I question may look as follows:

```
 <<problemI;
  a={10:99};
  b={10:99};
  c=47+a+b;
  answer('$%d$',c);
@2,0; a,b;
Find the sum of numbers 47, %d and %d.
>>
```

Here `a={10:99};` is a Spike-command which assigns to the variable `a` an integer value chosen randomly from the interval 10..99. In general, if one gives in a Spike-command any list of numbers inside a pair of curved brackets `{...}` written in MATLAB format, then Spikě replaces that pair of curved brackets by a number randomly chosen from the list. For example, an operator of the form

```
a = [{3,5,7,11}, {10:-2:2}]
```

in one run will be replaced by something like `a=[7,10]`, in another run the same operator will be replaced by, say, `a=[3,8]` etc. Only after this the processed Spikě operator will be given to MATLAB's processor. For example, one can use an assignment operator such as `a=10{3:7}8;` with a result of, say, `a=1068`.

The command `c=47+a+b;` solves the question in the sense that it finds the correct answer. Every type IR question must also have "the answer command"; in the above example this is the command

```
answer('$%d$',c);
```

Spikě replaces this command by the MATLAB command

```
answer=sprintf('$%d$',c);
```

before feeding it to MATLAB's processor. The meaning of this command should be clear to MATLAB users. Thus, the answer command produces a string variable `answer` which is later printed as one of the multiple choice answers.

Apart of this, a type IR question also produces four alternative incorrect answers. This is where the difference between type I and type R questions comes in. To produce incorrect answers Spikě runs the command part of type I question four more times. Spikě makes sure that all the five multiple choice answers are different. If the next answer is not "new" then Spikě runs the command part again until a new answer is produced. If Spikě fails to produce a new answer after a certain number of attempts, it stops and gives an error message.

Finally, Spikě randomly permutes the five multiple choice answers, but it does not permute the answers if the question is of type R, since in this case the multiple choice answers are naturally random.

Another example of a type I question.

```
<<text;
\bigskip
\noindent {\bf The Central Limit Theorem}
>>;
<<problemI;
  n={101:300};
  k=floor(7*n/15) + randi(round(n/15));
  z=(2*k/n-1)/sqrt(n);
  answer('%.4f',normcdf(z,0,1));
@2; n,k;
A fair coin is thrown %d times.
Find the probability that the number of tails will not exceed %d.
>>
```

This example consists of two Spikě-blocks: text-block and problemI-block. The content of a text-block is a piece of LaTeX-code which is executed by Spikě as it is. A text-block may contain some instructions for students on how to solve the problem or it may contain a title, etc.

The following is what the code above produces:

```
\bigskip
\noindent {\bf The Central Limit Theorem}

\problemtype{I}
\problA{E}{2}
A fair coin is thrown 161 times.
Find the probability that the number of tails will not exceed 86.

\medskip\noindent
(A) \ 0.4992 \quad
(B) \ 0.4993 \quad
(C) \ 0.5002 \quad
(D) \ 0.4984 \quad
(E) \ 0.5021 \quad
```

The Central Limit Theorem

7. [2m] A fair coin is thrown 161 times. Find the probability that the number of tails will not exceed 86.

(A) 0.4992 (B) 0.4993 (C) 0.5002 (D) 0.4984 (E) 0.5021

In a type R question the answer is supposed to be a real number, such as 13.7834801 . . . . In this case Spikě rounds the answer according to instructions given in the command part and then replaces the last digit of the answer by four other digits of the same parity: if the last digit is odd (even) then Spikě replaces the last digit by odd (even) digits.

For example, if in the code above we replace `problemI` by `problemR` then the result will be different:

```
\problemtype{R}
\problA{C}{2}
A fair coin is thrown 123 times.
Find the probability that the number of tails will not exceed 59.

\medskip\noindent
(A) \ $0.4981$\quad
(B) \ $0.4983$\quad
(C) \ $0.4985$\quad
(D) \ $0.4987$\quad
(E) \ $0.4989$\quad
```

8. [2m] A fair coin is thrown 123 times. Find the probability that the number of tails will not exceed 59.

(A) 0.4981 (B) 0.4983 (C) 0.4985 (D) 0.4987 (E) 0.4989

In this case there are two advantages of type R questions compared to type I questions: guessing the correct answer by students becomes difficult and the code runs faster.

Let's consider another example of a question of type R.

Question 1. *Prevalence of a medical condition in a population is equal to 0.15. What is the probability that exactly 3 people from a random sample of 9 people chosen from this population have the condition?*

There are three elements of this question which can be randomized: prevalence, the size of random sample and the number of people from the sample having the condition. We have to decide how we want them to randomize.

The following Spikě-code achieves this aim:

```
<<problemR;
  n={7:11};
  k={2:5};
  prvl={9:28}/100;
  p=nchoosek(n,k)*prvl^k*(1-prvl)^(n-k);
  answer('%.5f',p);
@3; prvl,k,n;
Prevalence of a medical condition in a population is equal to %.2f.
What is the probability that exactly %d people from a random sample
of %d people chosen from this population have the condition?
>>
```

Here the first three commands randomize the three elements of the question, the fourth command finds the correct answer and the last command tells Spikě to round the answer to five digits after the decimal point.

This is the result:

```
\problemtype{R}
\problA{D}{3}
Prevalence of a medical condition in a population is equal to 0.22.
What is the probability that exactly 3 people from a random sample
of 8 people chosen from this population have the condition?

\medskip\noindent
(A) \ $0.17210$ \quad
(B) \ $0.17212$ \quad
(C) \ $0.17214$ \quad
(D) \ $0.17216$ \quad
(E) \ $0.17218$ \quad
```

9. [3m] Prevalence of a medical condition in a population is equal to 0.22. What is the probability that exactly 3 people from a random sample of 8 people chosen from this population have the condition?

(A) 0.17210 (B) 0.17212 (C) 0.17214 (D) 0.17216 (E) 0.17218

Everything will work in the same way if we replace the identifier `problemR` by `problemI` but in this case the incorrect answers will be different.

3.5. **Volume of questions of type I and R.** With every question of type I or R we can associate a positive integer which we shall call *random volume* or just *volume* of the question. By definition, the volume of a question of type IR is the power of the set of all possible different questions which Spikě can produce using the question's code. There is also a second and different way to define the volume: it is the power of the set of all possible different answer strings which Spikě can produce using the questions code. Often there is no difference between these two volumes but in general they are different. To distinguish them, we shall call the second volume the *answer volume*. In any case the volume is always larger than or is equal to the answer volume.

Let's consider an example.

```
 <<problemI;
  a={1:6};
  b={1:6};
  c=a+b;
  answer('$%d$',c);
@2,0; a,b;
Find the sum of numbers %d and %d.
>>
```

The volume of this question is 36 since each of the variables a and b can take 6 different values and both these variables are used in the question's text. But the set of all possible answers is $2, 3, \ldots, 12$, and hence the answer volume is 11. It is desirable to keep volumes of questions of type IR large enough, say, at least 200. This is especially important for questions of type I since very small volumes may slow down processing of spike-code considerably.

3.6. **Format of the text part of questions of type I and R.** The text part of a question of type IR is what comes between at-line and the closing spike-bracket `>>`. The text part is a plain text written in LaTeX format with one exception: the percentage symbol `%` has a special role. In the text part the percentage symbol is used as a formatting operator in exactly the same way as in Matlab functions `fprintf` and `sprintf`. But the percentage symbol should **not** be the first character of a line in the text part, if it is to be used as a formatting operator, since any line in the text part which starts with percentage symbol is considered by Spikě as a comment and is therefore ignored. If a line starts with `%` as a formatting operator then just place the space character before `%`.

For example, the following spike-block will generate an error message.

```
<<problemI;
  a={1:6};
  b={1:6};
  c=a+b;
  answer('$%d$',c);
@2,0; a,b;
Find the sum of numbers
%d and %d.
>>
```

To fix the error one can place a space character before `%d` or move `%d` to previous line:

```
<<problemI;
  a={1:6};
  b={1:6};
  c=a+b;
  answer('$%d$',c);
@2,0; a,b;
Find the sum of numbers
 %d and %d.
>>
```

or

```
<<problemI;
  a={1:6};
  b={1:6};
  c=a+b;
  answer('$%d$',c);
@2,0; a,b;
Find the sum of numbers %d
and %d.
>>
```

3.7. **Questions of type T.** A type T question (T for "true") has the following structure:

```
<<problemT;
% False statements
False statement 1
False statement 2
...
```

```
False statement N
-------
% True statements
True statement 1
True statement 2
...
True statement M
@N1[,N2[,N3]]; k;
How many of the following assertions are true?
>>
```

Here Spikě chooses randomly `k` statements from the list of `N+M` statements, where the first `N` statements are false and the next `M` statements are true. True statements should follow after the false ones, and they must be separated by seven dash line `-------`. The lines which start with `%` are comments and are ingnored by Spike. Hence, it is not necessary to write `% False statements` and `% True statements`, but these lines are a good reminder.

Let's consider examples.

```
<<problemT;
% False statements
Antelope
Zebra
Giraffe
Rabbit
Hippopotamus
Elephant
-------
% True statements
Tiger
Wolf
Lion
Cheetah
Bobcat
Cat
Coyote
@2; 4;
How many of the following animals are carnivores?
>>
```

This is what Spikě produces:

```
\problemtype{T}
\problA{C}{2}
How many of the following animals are carnivores?

\begin{enumerate}
\item Cheetah
\item Hippopotamus
\item Rabbit
\item Zebra
\end{enumerate}

 (A) \ 4 \quad
 (B) \ 0 \quad
 (C) \ 1 \quad
 (D) \ 2 \quad
 (E) \ 3 \quad
```

10. [2m] How many of the following animals are carnivores?

(1) Cheetah
(2) Hippopotamus
(3) Rabbit
(4) Zebra

(A) 4 (B) 0 (C) 1 (D) 2 (E) 3

In the next run Spikě produces something else:

```
\problemtype{T}
\problA{C}{2}
How many of the following animals are carnivores?

\begin{enumerate}
\item Zebra
\item Cat
\item Coyote
\item Elephant
\end{enumerate}
```

11. [2m] How many of the following animals are carnivores?

(1) Zebra
(2) Cat
(3) Coyote
(4) Elephant

(A) 1 (B) 3 (C) 2 (D) 0 (E) 4

Another example.

```
<<text;
\bigskip
\noindent {\bf Properties of $E(X)$ and $\Var(X)$}
>>
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Properties of E(X) and Var(X)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
<<problemT;
% False
$x_1+\ldots+x_n=1.$
$E(XY) = E(X)E(Y).$
$E(X^2) = [E(X)]^2.$
$E(\abs{X}) = \abs{E(X)}.$
$E(g(X)) = g(E(X)).$
For any real number $c$ \ $\Var(cX) = c\Var(X).$
If $c$ is a constant then $E(c) = 0.$
If $c$ is a constant then $\Var(c) = c.$
$E(X) \geq 0.$
-----------------------------------------------------------------
% True
$p_1+\ldots+p_n=1.$
$E(X+Y) = E(X)+E(Y).$
For any number $c$ \ $E(cX) = cE(X).$
If $c$ is a constant then $E(c) = c.$
$E(X) = p_1x_1+\ldots+p_nx_n.$
$E(X^2) = p_1x_1^2+\ldots+p_nx_n^2.$
$E(g(X)) = p_1g(x_1)+\ldots+p_n g(x_n).$
$\Var(cX) = c^2\Var(X).$
If $X$ and $Y$ are independent then $E(XY) = E(X)E(Y).$
If $X$ and $Y$ are independent then $\Var(X+Y) = \Var(X) + \Var(Y).$
For any real number $c$ \ $\Var(cX) = c^2\Var(X).$
$\Var(X) = E(X^2)- [(E(X)]^2.$
$\Var(X) \geq 0.$
$E(X^2) \geq 0.$
If $c$ is a constant then $\Var(c) = 0.$
If $\Var(X) = 0$ then $X$ is a constant random variable.
```

```
% ------------------------------------------------------------------
@1; 8;
Let $X$ be a random variable which takes values $x_1,\ldots,x_n$
and only these values with respective probabilities $p_1,\ldots,p_n,$
let $Y$ be another random variable and let $g(x)$ be a function.
How many of the following assertions are always correct?
>>
```

This is what Spikě produces in a run:

```
\bigskip
\noindent {\bf Properties of $E(X)$ and $\Var(X)$}

\problemtype{T}
\problA{C}{1}
Let $X$ be a random variable which takes values $x_1,\ldots,x_n$
and only these values with respective probabilities $p_1,\ldots,p_n,$
let $Y$ be another random variable and let $g(x)$ be a function.
How many of the following assertions are always correct?

\begin{enumerate}
\item If $c$ is a constant then $\Var(c) = 0.$
\item For any number $c$ \ $E(cX) = cE(X).$
\item $E(X^2) = [E(X)]^2.$
\item $\Var(X) = E(X^2)- [(E(X)]^2.$
\item $E(X) \geq 0.$
\item If $X$ and $Y$ are independent then $E(XY) = E(X)E(Y).$
\item $E(g(X)) = g(E(X)).$
\item $E(X^2) = p_1x_1^2+\ldots+p_nx_n^2.$
\end{enumerate}

 (A) \ 8 \quad
 (B) \ 1 \quad
 (C) \ 5 \quad
 (D) \ 6 \quad
 (E) \ 4 \quad
```

Properties of $E(X)$ and $\mathrm{Var}(X)$

12. [1m] Let X be a random variable which takes values $x_1, \ldots, x_n$ and only these values with respective probabilities $p_1, \ldots, p_n$, let Y be another random variable and let $g(x)$ be a function. How many of the following assertions are always correct?

(1) If c is a constant then $\mathrm{Var}(c) = 0$.
(2) For any number c $E(cX) = cE(X)$.
(3) $E(X^2) = [E(X)]^2$.
(4) $\mathrm{Var}(X) = E(X^2) - [(E(X)]^2$.
(5) $E(X) \geqslant 0$.
(6) If X and Y are independent then $E(XY) = E(X)E(Y)$.
(7) $E(g(X)) = g(E(X))$.
(8) $E(X^2) = p_1x_1^2 + \ldots + p_nx_n^2$.

(A) 8 (B) 1 (C) 5 (D) 6 (E) 4

3.8. **Questions of type G.** Questions of type G (G for "generate") are similar to questions of type T.

The structure of a type G question is this:

```
<<problemG;
  a_matlab_function_name
@N1[,N2[,N3]]; k;
How many of the following statements are true?
>>
```

Here `a_matlab_function_name` should be a MATLAB function which returns two variables: a string variable which contains a statement in LaTeX format and a false/true variable, which indicates whether the statement is false or true.

In a question of type T we create a bank of false and true statements related to a certain topic. But imagine, for example, that we want to test whether a student has learnt definition of congruence modulo an integer. We may ask him/her a question of this kind: "Is it true that $7 \equiv 38 \pmod{12}$." And we can create a bank of false and true statements of this kind for a type T question. There is another way to do this: write a MATLAB function, say `GetTwoCongruentIntegers` which returns two variables: a string of the kind $7 \equiv 38 \pmod{12}$ but with randomly chosen numbers and a false/true value which shows whether the randomly generated statement is false or true. Once this is done we can create a type G question as follows:

```
<<problemG;
  GetTwoCongruentIntegers
@1; 4;
How many of the following congruences are true?
>>
```

Spikě will run the MATLAB function `GetTwoCongruentIntegers` four times (as indicated in the second parameter of the `@`-line) to produce four statements counting the number of true statements along the way.

This is what this spike-code produces:

```
\problemtype{G}
\problA{D}{1}
How many of the following congruences are true?

\begin{enumerate}
\item $\congruent{108}{101}{7}$
\item $\congruent{88}{152}{8}$
\item $\congruent{31}{69}{10}$
\item $\congruent{63}{75}{3}$
\end{enumerate}

 (A) \ 2 \quad
 (B) \ 0 \quad
 (C) \ 1 \quad
 (D) \ 3 \quad
```

13. [1m] How many of the following congruences are true?

(1) $108 \equiv 101 \pmod{7}$
(2) $88 \equiv 152 \pmod{8}$
(3) $31 \equiv 69 \pmod{10}$
(4) $63 \equiv 75 \pmod{3}$

(A) 2 (B) 0 (C) 1 (D) 3

To get an idea of what's going on let us consider one silly example.

Write a MATLAB function `GetEvenInteger` which returns an even or an odd integer with equal probability:

```
function [s,b]=GetEvenInteger
% Returns a random integer s; b is true, if s is even
% and b is false if otherwise.

x=100+ 2 * randi(100);
b=(rand<0.5);
if ~b, x=x+1; end
s=sprintf('$%d$',x);
end
```

Then a run of the spike-code

```
<<problemG;
 GetEvenInteger
@1; 5;
How many of the following integers are even?
>>
```

will produce the following output:

```
\problemtype{G}
\problA{E}{1}
How many of the following integers are even?

\begin{enumerate}
\item $120$
\item $170$
\item $179$
\item $271$
\item $107$
\end{enumerate}

 (A) \ 1 \quad
 (B) \ 4 \quad
 (C) \ 3 \quad
 (D) \ 5 \quad
 (E) \ 2 \quad
```

14. [1m] How many of the following integers are even?

(1) 120
(2) 170
(3) 179
(4) 271
(5) 107

(A) 1 (B) 4 (C) 3 (D) 5 (E) 2

3.9. **Questions of type H.** Questions of type H are very similar to questions of type G, but provide more flexibility.

Let us consider the previous problem: we wish to test whether a student has learnt definition of congruence of two integers. In this case question number 10 from previous subsection will solve this problem. But imagine that we also wish to test whether a student has learnt definition of congruence class of an integer. For example, we may ask a few questions of this type:

$$\text{Is it true that } 17 \in [32]_4$$

or

$$\text{Is it true that } [17]_4 = [32]_4.$$

The thing here is that using the same three integers, say a, b and n, we can produce many equivalent statements which are true or false at the same time, for example, $a \equiv b \pmod{n}$, $[a]_n = [b]_n$, $a \in [b]_n$, etc. In this case it would be inconvenient to write separate MATLAB functions for three or more questions of this kind if we use type G question.

In type H question the command part has to produce two variables: `out` and `tf` (for true/false). The first variable `out` is a string variable which contains in LaTeX format the statement of the question and the second variable `tf` is a boolean variable which indicates whether this statement is true or not.

The structure of a type H question is this:

```
<<problemH;
  (Command part)
@N1[,N2[,N3]]; k;
How many of the following statements are true?
>>
```

The command part has to produce two variables: `out` and `tf`. Spikě runs the command part `k` times to generate `k` statements counting along the way the number of true statements produced.

Let us consider an example. Assume that we have written a MATLAB function `GetCongruentInts` which returns four variables: three random integer variables `a`, `b` and `n` and one boolean variable `tf`, which has value "true" if and only if `n` divides `a-b`. Then the aim of question 10 can be achieved by the following code:

```
<<problemH;
  [a,b,n,tf]=GetCongruentInts;
  out=sprintf('$\\congruent{%d}{%d}{%d}$',a,b,n);
```

```
@1;  4;
How many of the following statements are true?
>>
```

Here we have to write double backslash \\ according to usage of MATLAB's `sprintf` function. The LaTeX command `\congruent` can then be defined as one wishes, for example,
`\newcommand{\congruent}[3]{#1 \equiv #2 \ \ (\,\mathrm{mod} \ #3)}`.

This definition can be inserted into the top-file.

Now using the same function `GetCongruentInts` we can create other questions. For example,

```
<<problemH;
  [a,b,n,tf]=GetCongruentInts;
  out=sprintf('$[%d]_{%d}=[%d]_{%d}$',a,n,b,n);
@1;  4;
How many of the following statements are true?
>>

<<problemH;
  [a,b,n,tf]=GetCongruentInts;
  out=sprintf('$%d \\in [%d]_{%d}$',a,b,n);
@1;  4;
How many of the following statements are true?
>>

<<problemH;
  [a,b,n,tf]=GetCongruentInts;
  out=sprintf('$[%d]_{%d} \\cap[%d]_{%d} = \\emptyset$',a,n,b,n);
@1;  4;
How many of the following statements are true?
>>
```

This is the result:

```
\problemtype{H}
\problA{D}{1}
How many of the following statements are true?
\begin{enumerate}
\item $[106]_{4}=[138]_{4}$
\item $[33]_{13}=[181]_{13}$
\item $[56]_{3}=[53]_{3}$
\item $[119]_{10}=[138]_{10}$
\end{enumerate}

(A) \ 0 \quad
(B) \ 3 \quad
(C) \ 4 \quad
(D) \ 2 \quad
(E) \ 1 \quad

\problemtype{H}
\problA{C}{1}
How many of the following statements are true?
\begin{enumerate}
\item $44 \in [132]_{7}$
\item $112 \in [180]_{5}$
\item $113 \in [146]_{3}$
\item $35 \in [178]_{13}$
\end{enumerate}

(A) \ 3 \quad
(B) \ 0 \quad
(C) \ 2 \quad
(D) \ 1 \quad
```

```
(E) \ 4 \quad

\problemtype{H}
\problA{C}{1}
How many of the following statements are true?
\begin{enumerate}
\item $[78]_{4} \cap[114]_{4} = \emptyset$
\item $[61]_{3} \cap[98]_{3} = \emptyset$
\item $[115]_{3} \cap[124]_{3} = \emptyset$
\item $[115]_{9} \cap[178]_{9} = \emptyset$
\end{enumerate}

(A) \ 2 \quad
(B) \ 1 \quad
(C) \ 3 \quad
(D) \ 0 \quad
(E) \ 4 \quad
```

15. [1m] How many of the following statements are true?

(1) $[106]_4 = [138]_4$
(2) $[33]_{13} = [181]_{13}$
(3) $[56]_3 = [53]_3$
(4) $[119]_{10} = [138]_{10}$

(A) 0 (B) 3 (C) 4 (D) 2 (E) 1

16. [1m] How many of the following statements are true?

(1) $44 \in [132]_7$
(2) $112 \in [180]_5$
(3) $113 \in [146]_3$
(4) $35 \in [178]_{13}$

(A) 3 (B) 0 (C) 2 (D) 1 (E) 4

17. [1m] How many of the following statements are true?

(1) $[78]_4 \cap [114]_4 = \varnothing$
(2) $[61]_3 \cap [98]_3 = \varnothing$
(3) $[115]_3 \cap [124]_3 = \varnothing$
(4) $[115]_9 \cap [178]_9 = \varnothing$

(A) 2 (B) 1 (C) 3 (D) 0 (E) 4

3.10. **Questions of type V.** Finally, there are questions of type V (V for "verbatim").

A question of type V is not a random question. This type of question is introduced just in case one wants to include a non-randomized question into an assignment (why not?). The reason may be that not all questions can be randomized.

The structure of a type V question is as follows:

```
<<problemV;
@N1[,N2[,N3]]; X;
Write here whatever you want
but in LaTeX format.
>>
```

Here `X` is the correct answer to the question. Since this is a non-randomized question, the correct answer to it is known, and it should be indicated in the `@`-line as the third parameter. This is necessary since Spikě keeps all correct answers for marking purposes.

Let us consider an example.

```
<<problemV;
@2,1; D;
Which of the following chemical elements is a metal?

 (A) \ $O$ \quad  (B) \ $C$ \quad  (C) \ $H$ \quad  (D) \ $K$ \quad  (E) \ $N$
```

```
>>
```

This is what Spikě produces:

```
\problB{D}{2}{1}
Which of the following chemical elements is a metal?
\medskip

 (A) \ $\mathrm O$ \quad  (B) \ $\mathrm C$ \quad  (C) \ $\mathrm H$
 \quad  (D) \ $\mathrm K$ \quad  (E) \ $\mathrm N$
```

18. [2m,−1m] Which of the following chemical elements is a metal?

(A) O (B) C (C) H (D) K (E) N

The following is an actual type V question from one of the Algebra assignments.

```
<<problemV;
@2,2; C;
Let $X$ be any non-empty set and let $P(X)$ be the family of all subsets of $X.$
We endow the set $P(X)$ with binary operations of union $\cup$ as addition
and intersection $\cap$ as multiplication of sets.
Thus obtained triple $(P(X), \cup, \cap)$ is not a commutative ring with identity,
since it fails to meet one of the axioms (R1)--(R8)
of a commutative ring $(R,+,\cdot)$ with identity, given below.
Which axiom is that?
\begin{enumerate}
  \item[(R1) \ ] $\forall a,b \in R \ \ a+b= b+a;$
  \item[(R2) \ ] $\forall a,b,c \in R \ \ (a+b)+c= a+(b+c);$
  \item[(R3) \ ] $\exists 0_R \in R \ \forall a \in R \ \ a+0_R = a$ and $0_R+a=a;$
  \item[(R4) \ ] $\forall a \in R \ \exists b \in R \ \ a+b=0_R$ and $b+a=0_R.$
  \item[(R5) \ ] $\forall a,b,c \in R \ \ (a+b)c= ac+bc$ \quad and \quad $c(a+b) = ca+cb.$
  \item[(R6) \ ] $\forall a,b,c \in R \ \ (ab)c= a(bc).$
  \item[(R7) \ ] $\exists 1_R \in R \ \forall a \in R \ \ a\cdot 1_R = 1_R\cdot a=a.$
  \item[(R8) \ ] $\forall a,b \in R \ \ ab= ba.$
\end{enumerate}
\par
(A) \ (R2) \qquad  (B) \ (R3) \qquad (C) \ (R4) \qquad (D) \ (R5)
\qquad (E) \ (R6) \qquad (F) \ (R7)     % correct answer is (C)
>>
```

19. [2m,−2m] Let X be any non-empty set and let $P(X)$ be the family of all subsets of X. We endow the set $P(X)$ with binary operations of union $\cup$ as addition and intersection $\cap$ as multiplication of sets. Thus obtained triple $(P(X), \cup, \cap)$ is not a commutative ring with identity, since it fails to meet one of the axioms (R1)–(R8) of a commutative ring $(R, +, \cdot)$ with identity, given below. Which axiom is that?

(R1) $\forall a, b \in R \ \ a + b = b + a$;
(R2) $\forall a, b, c \in R \ \ (a + b) + c = a + (b + c)$;
(R3) $\exists 0_R \in R \ \forall a \in R \ \ a + 0_R = a$ and $0_R + a = a$;
(R4) $\forall a \in R \ \exists b \in R \ \ a + b = 0_R$ and $b + a = 0_R$.
(R5) $\forall a, b, c \in R \ \ (a + b)c = ac + bc$ and $c(a + b) = ca + cb$.
(R6) $\forall a, b, c \in R \ \ (ab)c = a(bc)$.
(R7) $\exists 1_R \in R \ \forall a \in R \ \ a \cdot 1_R = 1_R \cdot a = a$.
(R8) $\forall a, b \in R \ \ ab = ba$.

(A) (R2) (B) (R3) (C) (R4) (D) (R5) (E) (R6) (F) (R7)

This is a good question, but it is difficult to randomize it.

3.11. **Questions of type Z.** A question of type Z is not a question. If a spike-block `<<...>>` starts with `<<problemZ;` then Spikě ignores the whole spike-block. This feature is convenient if you wish to remove temporarily one or more questions from the assignment, or some question is not ready yet etc.

3.12. **Top-file.** Spikě creates an output LaTeX-file which contains one assignment for each student enrolled in a topic. In the beginning of this output file it is desirable to give some information relevant to the assignment such as: (1) assignment number, (2) due date, (3) the percentage of the total assessment which this assignment constitutes, (4) instructions on how to submit answers to the assignment, (5) what to do in certain exceptional cases, etc.

All this information should or can be given in the top-file. The top-file should or can contain not only assignment information but also LaTeX preamble commands and macro-commands. An example of a top-file is given below.

The command `%EOF` means end of file: Spikě ignores everything which follows after this line. This command is not compulsory.

Top-file consists of two parts: the MATLAB part and the LaTeX-part. The line which starts with `\documentclass` separates the MATLAB and the LaTeX-parts of the top-file. Spike executes commands in the MATLAB part and copies the LaTeX-part into the beginning of the output LaTeX-file.

Some of the LaTeX-part commands may contain references to the topic code and the assignment number, which are given as arguments to Spikě. In Spikě there are two LaTeX-commands `\topiccode` and `\NAss` which define the topic code and the assignment number respectively. Spikě automatically defines these commands when it encounters the line `\begin{document}` in the top-file.

That is, in the top-file one can use LaTeX commands `\topiccode` and `\NAss` after `\begin{document}` without defining them.

3.13. **Spikě variables.** Spike has several variables: `duedate, rseed, NAltAns, NRepeat. rseed` is a seed for MATLAB's generator of pseudo-random numbers, `NAltAns` is the number of alternative answers, `NRepeat` is the number of times each assignment question is repeated.

By default, `NAltAns`=5, and `NRepeat`=1. These default values can be changed in the MATLAB part of the top-file. They can also be changed in a data-block of spk-file.

For example, to change the number of alternative answers from five to four one can to insert the command

```
NAltAns = 4;
```

to the top-file. Another option is to insert the following data block in the beginning of spike-file:

```
<<data;
  NAltAns = 4;
>>
```

3.14. **Assignment cycle.** The code which is given in a spike-file is executed in a cycle. We shall call it *assignment cycle*. To produce k assignments Spikě runs the code of spike-file k times. How many times to run the spike-code is determined by the student-list file: if there are, say, 27 student ID's before the first seven dashes line, then the spike-code will be run 27 times. As a result 27 assignments will be prepared. They are all given in a single LaTeX file, so a student can see other students assignments too.

It should be mentioned that Matlab-code which is given in the top-file is executed only once and before preparing the first assignment. On the other hand, any Matlab code which is given in data-block of a spike-file is executed in every run of the assignment cycle. Thus, if for example the number of alternative answers is set to four by the command `NAltAns = 4;` in the top-file, but somewhere in spike-file it is changed to another value, say `NAltAns = 3;`, then this value will stay until it is changed by another command. The same remark applies to all other spike variables, such as `NRepeat` etc.

3.15. **Curved brackets operator.** It was already mentioned that Spikě has a special feature which allows to randomize variables. Since the aim of Spikě is to create randomized questions, randomization operators in Spikě are obviously used very often. While one can use Matlab's operators `rand` and `randi`, Spike's randomization operators provide a little bit more flexibility.

For example, assume that we wish a variable `a` to take values from 5 to 12 with equal probabilities. This can be achieved by the following command:

```
a = {5:12};
```

If we want `b` to take values from the interval 2..5 or from the interval 10..15 so that every number has equal chance to be chosen, then we can use the operator

```
a = {2:5,10:15};
```

One can have any number of intervals here. Moreover, the intervals may overlap:

```
a = {1:5,7:9, 10:14, 12:16};
```

But if the intervals overlap, the overlapped numbers have a better chance to be chosen. In general, probability to be chosen in a curved bracket operator is proportional to the number of appearances of the number in the list between curved brackets. This can be used to randomize certain variables in non-uniform way. For instance, assume that we wish a variable `c` to be equal to 3 with two chances out of five and to be equal to 7 with three chances out of five. This can be achieved by this operator:

```
c = {3,3,7,7,7};
```

One can use more than one pair of curved brackets in one command. For example,

```
x = {1:5} + {3:6};
```

This means that `x` will be given a value of a random variable which is the sum of two variables, first taking values from 1..5 and the second taking values from 3..6, with equal probabilities. The operator

```
x = [{-3:5}, {3:6}, {-9:-2}];
```

would return a random vector. The curved brackets can be nested. For example, one can use the operator

```
t = {1{2:5}:3{3:5}};
```

In such a case Spikě first processes inner pairs of curved brackets.

To avoid problems it is good to know how Spikě processes commands with curved brackets. Spike first looks for curved brackets; it finds the first innermost pair `{some stuff}` of curved brackets within a line and then it replaces it with a randomly chosen element of the Matlab array `[stuff]` so that every element of the array has an equal chance to be chosen. So, if some element is repeated three times then it has three times more chance to be chosen. Once the first innermost pair of curved brackets is processed, Spike proceeds to the second innermost pair and so on. Once all curved brackets are replaced, the line is given to Matlab's processor.

For example, the command `t = {1{2:5}:3{3:5}};` may go through the following stages:

```
t = {1{2:5}:3{3:5}};  -->  t = {13:3{3:5}};  -->  t = {13:35};  -->  t = 22;
```

So, Matlab's processor will receive the command `t = 22;`

For example, the following Spikě code

```
a = {1:8};
b = {1:8};
while a==b, b={1:8}; end
```

should not be used. The aim of this code is to produce two different random integers `a` and `b`. But if it turns out that after the execution of the first two commands values of `a` and `b` are the same, say, 5, and the command `b={1:8};` will again assign the value 5 (just by chance) to `b` then we will get an infinite loop. This is because in all cycles the command `b={1:8};` will in fact be `b=5;` To avoid the infinite loop, one should not use curved brackets:

```
a = {1:8};
b = {1:8};
while a==b, b=1+randi(7); end
```

In creating randomized questions it is often necessary to obtain two or more different random numbers. In this regard, there are good news. For example, if we wish to get two different random integers `a` and `b` with values from the interval 1..8 then we can use the command

```
a,b!={1:8};
```

Here before the assignment operator `!=` we can give a list of variables separated by commas. For each variable `x` in the list the assignment operator `x={1:8}` will be executed in such a way that all variables in the list will get different random values. It is often desirable to have different values of variables which are sorted. In this case one should use `!<` instead of `!=`. For instance, the command

```
a,b,c,d!={1:20};
```

will assign different values to variables `a,b,c,d` in such a way that `a<b<c<d.`

The operators `!=` and `!<` we shall call multiple assignment operators.

The only restriction in the usage of multiple assignment operators is that there should be only one pair of curved brackets and nothing else. For example, the operators

```
a,b,c,d!={1:7, 2:10, 13, 14};
x,y,z!<{-0.1:0.7, 2:2:10};
```

are ok, but the operators

```
  a,b,c,d!={1:7, 2:10, 1{3:4}};
  x,y,z!<{-1:7}/10;
```

are not.

Finally, the array inside curved brackets does not need to be a number array. For instance,

```
  a,b!={'A'..'Z'};
  x,y,z!<{'a'..'k', 'hello'};
```

would work. In the second case here the three characters will be sorted in alphabetical order.

4. At-line and its parameters

Spike-blocks, except data-blocks and text-blocks, have what we shall call *at-line* or `@`-line. At-line separates the command-part of a spike-block from the text-part. But apart of this, at-lines have arguments which are separated by semi-colons. The first two arguments are compulsory, the third one is optional.

4.1. **First argument of at-line: marks.** In the first argument we can give from one to four numbers such as in

```
@3,2; a,b,c;
```

This at-line has two arguments and the first argument has two numbers. The first number is treated as the number of points which a student gets for a correct answer to the question; the second number, if there is one, is treated as the number of negative "punishment" points which a student gets for an incorrect answer to the question; the third number, if there is one, is treated as the number of negative "punishment" points which a student gets for not answering the question. Finally, there can also be an optional fourth number in the first argument interpretation of which is up to a particular user.

Depending on the number of numbers in the first argument of at-line, Spikě prints different LaTeX-commands into the output file. If there is one number `N1` in the first argument of at-line, then the LaTeX command printed into the output file will be `\problA`. If there are two (respectively, three, four) numbers `N1,N2` (respectively, `N1,N2,N3` or `N1,N2,N3,N4`) in the first argument of at-line, then the LaTeX command printed into the output file will be `\problB` (respectively, `\problC`, `\problD`).

For example, if we are testing knowledge of times-table, then we can use the following spike-block:

```
<<problemI;
  a = {2:9};
  b = {2:9};
  answer('$%d$',a*b);
@2; a,b;
Find the product of numbers %d and %d.
>>
```

with the following result:

```
\problemtype{I}
\problA{B}{2}
Find the product of numbers 5 and 9.

\medskip\noindent
(A) \ $32$ \quad
(B) \ $45$ \quad
(C) \ $28$ \quad
(D) \ $12$ \quad
(E) \ $72$ \quad
```

20. [2m] Find the product of numbers 5 and 9.

(A) 32 (B) 45 (C) 28 (D) 12 (E) 72

The part [2m] of this text appears here because the command `\problA` was defined (in the top-file) to show the number of points which a student gets for the correct answer. If we wish to penalize a student for an incorrect answer, then we presumably need to show in the question itself what the penalty is. The penalty for an incorrect answer should be given as the second number of the first at-line argument, for example,

```
<<problemI;
  a = {2:9};
  b = {2:9};
  answer('$%d$',a*b);
@2,2; a,b;
Find the product of numbers %d and %d.
>>
```

Since there are two numbers in the first argument, Spikě will use `\problB` LaTeX command.

```
\problemtype{I}
\problB{D}{2}{2}
Find the product of numbers 2 and 3.

\medskip\noindent
(A) \ $12$ \quad
(B) \ $35$ \quad
(C) \ $36$ \quad
(D) \ $6$ \quad
(E) \ $27$ \quad
```

21. [2m,−2m] Find the product of numbers 2 and 3.

(A) 12 (B) 35 (C) 36 (D) 6 (E) 27

The part [2m,−2m] of this text appears here because the command `\problB` was defined (in the top-file) to show the number of points which a student gets for the correct answer and for an incorrect answer.

Finally, if you are a teacher cruel enough to penalize students for no answer too, then you can use three numbers such as here:

```
<<problemI;
  a = {2:9};
  b = {2:9};
  answer('$%d$',a*b);
@2,2,1; a,b;
Find the product of numbers %d and %d.
>>
```

in order to get this:

```
\problemtype{I}
\problC{E}{2}{2}{1}
Find the product of numbers 6 and 5.

\medskip\noindent
(A) \ $49$ \quad
(B) \ $54$ \quad
(C) \ $32$ \quad
(D) \ $15$ \quad
(E) \ $30$ \quad
```

How to define the LaTeX command `\problC` in the top-file is up to you; I do not have a definition for this command since I am a kind teacher.

Finally, there can be a fourth number in the first argument with resulting LaTeX command `\problD`, but interpretation and usage of this fourth number is open.

4.2. **Second argument of at-line: the text part arguments.** The second argument of at-line shows the list of variables which are used in the text-part of spike-block.

Spike is about randomizing mathematical problems. When one chooses a problem to be randomized by Spike, one has to decide which and how many elements of the problem is to randomize. Once the elements to be randomized are chosen, in the second argument of the at-line we give a list of variables values of which are supposed to replace to be randomized elements, and these elements themselves in

the text of the problem we replace by the Matlab formatting operators such as `%d, %.3f, %c, %s` etc, exactly as they are used in Matlab's `fprintf` command.

For example, let us consider the following problem:

Find the scalar product of vectors $\mathbf{a} = (3, -2, 5)$ *and* $\mathbf{b} = (6, 4, -4)$.

Here we can randomize two vectors {(} 3, −2, 5) and {(} 6, 4, −4) (or, if you like, six numbers). The following Spikě code will do this:

```
<<problemI;
  a=randi(20,1,3)-10;
  b=randi(20,1,3)-10;
  answ=a*b';
  answer('$%d$',answ);
@1,4; a(1),a(2),a(3), b(1),b(2),b(3);
Find the scalar product of vectors $\mathbf a = (%d,%d,%d)$ and
$\mathbf b = (%d,%d,%d).$
>>
```

```
\problemtype{I}
\problB{C}{1}{4}
Find the scalar product of vectors $\mathbf a = (8,4,-7)$ and
$\mathbf b = (2,-6,9).$

\medskip\noindent
(A) \ $4$ \quad
(B) \ $0$ \quad
(C) \ $-71$ \quad
(D) \ $8$ \quad
(E) \ $16$ \quad
```

22. [1m,−4m] Find the scalar product of vectors $\mathbf{a} = (8, 4, -7)$ and $\mathbf{b} = (2, -6, 9)$.

(A) 4 (B) 0 (C) −71 (D) 8 (E) 16

(Since this question is something really very basic the penalty score is big). This looks ok, but if we have, say, two 7-dimensional vectors then this solution would be a bit clumsy. In this case one may wish to write a Matlab function which tranforms a vector or a matrix into LaTeX format. Assume that we have written such a function and named it `sLaTeXMatrix`. Then the following Spikě code can be used:

```
<<problemI;
  a=randi(20,1,7)-10;
  b=randi(20,1,7)-10;
  sa=sLaTeXMatrix(a);
  sb=sLaTeXMatrix(b);
  answ=a*b';
  answer('$%d$',answ);
@1,4; sa, sb;
Find the scalar product of vectors
$$\mathbf a = %s$$
and
$$\mathbf b = %s.$$
>>
```

Note that the formatting operator used here is `%s` since `sa` and `sb` are string variables.

```
\problemtype{I}
\problB{A}{1}{4}
Find the scalar product of vectors
$$\mathbf a = \left(\begin{matrix}5&9&0&-7&1&-3&-2\\\end{matrix}\right)$$
and
$$\mathbf b = \left(\begin{matrix}-1&-1&-8&-9&3&-5&5\\\end{matrix}\right).$$
```

```
\medskip\noindent
(A) \ $57$ \quad
(B) \ $34$ \quad
(C) \ $-33$ \quad
(D) \ $9$ \quad
(E) \ $-15$ \quad
```

23. [1m,−4m] Find the scalar product of vectors

$$\mathbf{a} = \begin{pmatrix} 5 & 9 & 0 & -7 & 1 & -3 & -2 \end{pmatrix}$$

and

$$\mathbf{b} = \begin{pmatrix} -1 & -1 & -8 & -9 & 3 & -5 & 5 \end{pmatrix}.$$

(A) 57 (B) 34 (C) −33 (D) 9 (E) −15

Finally, one may wish to randomize not only numbers in the above problem but also notation for vectors **a** and **b**. That is, we may wish to use different letters to denote these two vectors. The following code achieves this aim.

```
<<problemI;
  x,y!<{'abcdefgxyz'};
  a=randi(20,1,3)-10;
  b=randi(20,1,3)-10;
  answ=a*b';
  answer('$%d$',answ);
@1,4; x, a(1),a(2),a(3), y, b(1),b(2),b(3);
Find the scalar product of vectors $\mathbf %c = (%d,%d,%d)$ and
$\mathbf %c = (%d,%d,%d).$
>>
```

```
\problemtype{I}
\problB{B}{1}{4}
Find the scalar product of vectors $\mathbf d = (6,3,-4)$ and
$\mathbf g = (-9,5,10).$

\medskip\noindent
(A) \ $-84$ \quad
(B) \ $-79$ \quad
(C) \ $14$ \quad
(D) \ $-35$ \quad
(E) \ $-19$ \quad
```

24. [1m,−4m] Find the scalar product of vectors $\mathbf{d} = (6, 3, -4)$ and $\mathbf{g} = (-9, 5, 10)$.

(A) −84 (B) −79 (C) 14 (D) −35 (E) −19

Finally, one may randomize even the wording of the problem.

```
\problemtype{I}
\problB{D}{1}{4}
Find $\mathbf d \bullet \mathbf f$ if  $\mathbf d = (-9,0,2)$ and
$\mathbf f = (9,6,9).$

\medskip\noindent
(A) \ $-27$ \quad
(B) \ $-62$ \quad
(C) \ $-3$ \quad
(D) \ $-63$ \quad
(E) \ $-59$ \quad
```

25. [1m,−4m] Find $\mathbf{d} \bullet \mathbf{f}$ if $\mathbf{d} = (-9, 0, 2)$ and $\mathbf{f} = (9, 6, 9)$.

(A) -27 (B) -62 (C) -3 (D) -63 (E) -59

This was achieved by the following code:

```
<<problemI;
  x,y!<{'abcdefgxyz'};
  ss='the scalar product of vectors';
  bb=(rand<0.5);
  if bb, ss=sprintf('$\\mathbf %c \\bullet \\mathbf %c$ if ',x,y); end
  a=randi(20,1,3)-10;
  b=randi(20,1,3)-10;
  answ=a*b';
  answer('$%d$',answ);
@1,4; ss, x, a(1),a(2),a(3), y, b(1),b(2),b(3);
Find %s $\mathbf %c = (%d,%d,%d)$ and
$\mathbf %c = (%d,%d,%d).$
>>
```

In this case in 50 percent of cases the phrase "... the scalar product of vectors ..." will be used and in another 50 percent of cases Spikě will use an expression of the kind "... $\mathbf{a} \bullet \mathbf{b}$ if ...".

4.3. **Third argument of at-line:** NAltAns. Spike allows to change the number of multiple choice answers from default value of five to any number between 2 and 8. This can be done in a data-block directly by changing value of the spike variable `NAltAns`. For example, the spike code

```
<<data;
  NAltAns=2;
  p=5;
>>

<<problemI;
  [f,g]=GetIrreducibleDegree2Poly(p);
  x=rand;
  if x<0.5, ss=PolyToLaTeX(f); answ='Yes'; end
  if x>=0.5, ss=PolyToLaTeX(g); answ='No'; end
  answer('%s',answ);
@3,2; ss,p;
Is the polynomial $f(x) = %s$ from $\mbZ_{%d}[x]$ irreducible?
>>
```

gives this output:

```
\problemtype{I}
\problB{A}{3}{2}
Is the polynomial $f(x) = 2+x+x^{2}$ from $\mbZ_{5}[x]$ irreducible?

\medskip\noindent
(A) \ Yes \quad
(B) \ No \quad
```

26. [3m,−2m] Is the polynomial $f(x) = 2 + x + x^2$ from $\mathbb{Z}_5[x]$ irreducible?

(A) Yes (B) No

The problem is that once value of `NAltAns` is changed, it will affect all subsequent questions. Of course, the old value can be saved and then restored back after a problem but this is a bit clumsy.

If one wishes to change the number of alternative answers only in one question then this number can be given as the third argument of the at-line:

```
<<problemI;
  [f,g]=GetIrreducibleDegree2Poly(p);
  x=rand;
```

```
  if x<0.5, ss=PolyToLaTeX(f); answ='Yes'; end
  if x>=0.5, ss=PolyToLaTeX(g); answ='No'; end
  answer('%s',answ);
@3,2; ss,p;  2;
Is the polynomial $f(x) = %s$ from $\mbZ_{%d}[x]$ irreducible?
>>
\problemtype{I}
\problB{A}{3}{2}
Is the polynomial $f(x) = 1+3x+2x^{2}$ from $\mbZ_{5}[x]$ irreducible?

\medskip\noindent
(A) \ No \quad
(B) \ Yes \quad
```

27. [3m,−2m] Is the polynomial $f(x) = 1 + 3x + 2x^2$ from $\mathbb{Z}_5[x]$ irreducible?

(A) No (B) Yes

This way of changing the number of alternative answers will not affect other questions.

5. Marking

When Spikě processes a spike-file it not only produces the output assignment LaTeX-file, but it also keeps information about assignment questions: what type of question-block (I,R,T,G, H or V) it was created by, question number, the positive mark for the correct answer given to the question, the negative mark given for an incorrect answer and the correct answer to the question.

This information is written in the output LaTeX-file after the `\end{document}` command. LaTeX ignores everything which follows after `\end{document}`, and so this is safe to do so. Of course, this information could be kept in a separate file, but firstly it is desirable to decrease the number of files, and secondly it is helpful to keep correct answers to an assignment in the same file as the assignment itself in order to avoid a possible confusion.

I give a real-life example. In my PDE topic with topic code `MATH3712` there are sixteen students with student numbers (these numbers were modified)

```
2019463   2083123   2095869   2108070
2028028   2092052   2099265   2112299
2060521   2092462   2104714   2123762
2062027   2093747   2107353   2126266
```

These numbers are kept in the file `MATH3712.tex` in `MATH3712\` folder. Hence, Spikě creates 16 assignments, one for each student.

At the end of the output assignment LaTeX-file for the third assignment after the command `\end{document}` the following text is inserted:

```
Topic code:          MATH3712
Assignment number:   3
Number of questions: 22

PrbType:RRRRRRTRTTRRRRTRRRRRRR
%MARKING
Numbers:123456789012345678901 2
Specfcs:.....................
Weights:3322233211222122222312
Neg wts:00000010000001000100 00
2108070:ADCDEDCEDCADDACEDACEEA
2060521:EDBBBADEDCDECDACDADEAC
2092052:ECEEEECADDAEBBCBAAEABC
2092462:BDABEEDAAAACBEDADBADCD
2123762:DCBDDDAAACABCABCDABAEA
2083123:BACECACBAAAAADABDBBBDE
2107353:EDAEDBBACAEBBBCBAEBABD
2019463:DCABCECABABBBADCCECAED
2104714:AEDAEBBBABDDCBACDCEAEA
```

```
2099265:ECCDBACACBBDDACACBDADD
2126266:DEDEBBCAACDBDAABEBCDBA
2062027:EBCAEABAACBACBCDAEBEBB
2093747:BDBECDCACDADDDABABADDC
2028028:ADEDCBBACCDAEDDEBAEECC
2112299:BDDEAEABDBBDBCCBDCDCCA
2095869:ECCCDADCAACBBEDADEDBAD
-------
%Insert here students' answers.
```

Once students answers are received, I insert them after the seven dash line ------- as follows:

```
Topic code:          MATH3712
Assignment number:   3
Number of questions: 22

PrbType:RRRRRRTRTTRRRRTRRRRRRR
%MARKING
Numbers:123456789012345678901 2
Specfcs:.....................
Weights:3322233211222122222312
Neg wts:000000100000010001 0000
2062027:EBCAEABAACBACBCDAEBEBB
2028028:ADEDCBBACCDAEDDEBAEECC
2092052:ECEEEECADDAEBBCBAAEABC
2108070:ADCDEDCEDCADDACEDACEEA
2083123:BACECACBAAAAADABDBBBDE
2099265:ECCDBACACBBDDACACBDADD
2126266:DEDEBBCAACDBDAABEBCDBA
2112299:BDDEAEABDBBDBCCBDCDCCA
2107353:EDAEDBBACAEBBBCBAEBABD
2092462:BDABEEDAAAACBEDADBADCD
2019463:DCABCECABABBBADCCECAED
2060521:EDBBBADEDCDECDACDADEAC
2104714:AEDAEBBBABDDCBACDCEAEA
2093747:BDBECDCACDADDDABABADDC
2123762:DCBDDDAAACABCABCDABAEA
2095869:ECCCDADCAACBBEDADEDBAD
-------
2108070:ADCDEDCDDCADDABEDACEEA
2095869:ECCCDADCAACBBEDADEDBAD
2060521:EDBBBADEDCDECDACDAEEAC
2028028:ADEDCBBABBDAEDDEBAEECC
2083123:BACECACBAAAAADABDBBBDE
2093747:BDBECDCACDADDDABABADDC
2104714:AEDAEBBBABDDCBACDCEAEA
2019463:DBABEECAAABDBADCCCCAED
2099265:ECCDBACACBBDDACACBDADD
2107353:EDAEDBBACAEBBBCBAEBABD
2092462:BDABEEDADAACBEDADBADDD
2062027:EBCAEABAACBACBCDAEBEBB
2123762:CBADBE.ADBCCE.BCB.ACDB
```

Students ID's don't need to be sorted. The above information is used by a marking program `SpikeMark` to mark the assignment.

The following text is the marking results file prepared by `SpikeMark`.

Flinders University 2014 Semester 2
MATH3712 Assignment 3 marks as by 04-Nov-2014

	Stud.ID	Answers	points	mark
12	****027	+++++ +++++ +++++ +++++++	45	20.0
14	****028	+++++ +++CC +++++ +++++++	43	19.1
3	****052	....	0	0.0
1	****070	+++++ ++E++ ++++C +++++++	41	18.2
6	****123	+++++ +++++ +++++ +++++++	45	20.0
10	****265	+++++ +++++ +++++ +++++++	45	20.0
11	****266	....	0	0.0
15	****299	....	0	0.0
7	****353	+++++ +++++ +++++ +++++++	45	20.0
4	****462	+++++ +++A+ +++++ +++++C+	43	19.1
8	****463	+C++C +++B+ +B+++ ++E++++	34	15.1
2	****521	+++++ +++++ +++++ +++++++	45	20.0
9	****714	+++++ +++++ +++++ +++++++	45	20.0
13	****747	+++++ +++++ +++++ +++++++	45	20.0
5	****762	DCB+D D.+AC ABC.+ +D.BAEA	8	3.6
16	****869	+++++ +++++ +++++ +++++++	45	20.0
	Ques.N.	12345 67890 12345 6789012	Total:	Total:
	Weight	33222 33211 22212 2222312	45	20
	Neg.wt	00000 01000 00010 0010000		

If a student's answer is incorrect, then a correct answer `A,B,C,...` or `"-"` is shown in the table.

`"-"` means that the answer given by a student is incorrect and the correct answer will be given later.

`"+"` means that the answer given by a student is correct.

`"."` means that the answer was not given.

Numbers of correct answers given to each question:

```
      1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22
     12  11  12  13  11  12  12  12   9  11  12  11  12  12  12  13  12  11  11  12  11  12
%:   92  85  92 100  85  92  92  92  69  85  92  85  92  92  92 100  92  85  85  92  85  92
                                     --
```

Number of students enrolled in this topic: 16.
Number of submissions: 13.
The average mark for this assignment: 14.6.

6. Appendix A: Examples of assignments

The best way to learn to use Spikě is to follow an example.

6.1. **A statistics assignment.** Since statistics is a very popular subject I decided to create an example of a "statistics assignment". On second thought, medical applications of statistics are even more popular, and so many questions in this assignment have medical flavour. I did not test this assignment with real students and so I do not guarantee that this code is free of mistakes.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Definition of probability
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
<<problemI;
  N={2:12};
  if (N==2)||(N==12), p=1/36; end
  if (N==3)||(N==11), p=1/18; end
  if (N==4)||(N==10), p=1/12; end
  if (N==5)||(N==9), p=1/9; end
  if (N==6)||(N==8), p=5/36; end
  if N==7, p=1/6; end
  answer('$%s$',sLaTeXFrac(p));
@2,1; N;
Assume that a pair of fair dice are thrown.
What is the probability that the sum of numbers
on their faces is equal to %d?
>>

<<text;
\bigskip
\noindent {\bf Definition of probability.}
>>

<<problemR;
  N={7:20};
  R={8:22};
  a=zeros(N,N);
  for j=1:N, for k=1:N, if j^2+k^2 < R, a(j,k)=1; end; end; end;
  ans=sum(sum(a))/N^2;
  answer('%.3f',ans);
@2,1; N,N,R;
Assume that we choose randomly and independently two numbers~$X$ and~$Y$
between 1 and %d, so that each of those %d numbers have an equal
chance to be chosen and maybe be equal. What is the probability
that $X^2+Y^2$ is less than $%d?$
Round the answer to three digits.
>>


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Expectation E(X)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
<<text;
\bigskip
\noindent {\bf Expectation $E(X)$ of a random variable.}
>>

<<problemI;
  p=GetRandProbDist(4,1);
  X=randi(10,1,4);
  X=sort(X);
  X=X+[0,1,2,3];
  ans=sum(p.*X);
  answer('%0.1f',ans);
```

```
@2,0; X(1),X(2),X(3),X(4),p(1),p(2),p(3),p(4);
A random variable $X$ takes values $%d,%d,%d,$
and $%d$ with probabilities $%.1f,%.1f,%.1f,$ and $%.1f$
respectively. Find $E(X).$
>>


<<problemI;
  x=randi(900)/100;
  y=randi(900)/100;
  a=randi(90)/10;
  b=randi(90)/10;
  answer('%.4f',a*x+b*y);
@2,0; x,y,a,b;
Random variables $X$ and $Y$ have expectations $%.2f$ and
$%.2f$ respectively. What is the expectation of the random
variable $%.1fX+%.1fY?$
>>

<<text;
\bigskip
\noindent {\bf Variance $\Var(X)$ of a random variable.}
>>

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Variance Var(X)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
<<problemI;
  p=GetRandProbDist(3,1);
  X=randi(10,1,3);
  X=sort(X);
  X=X+[0,1,2];
  X2=X.*X;
  ans=sum(p.*X2)-sum(p.*X)^2;
  answer('%0.2f',ans);
@2,0; X(1),X(2),X(3),p(1),p(2),p(3);
A random variable $X$ takes values $%d,%d,$
and $%d$ with probabilities $%.1f,%.1f,$ and $%.1f$
respectively. Find variance of $X.$
>>

<<problemI;
  x=randi(900)/100;
  c=randi(90)/10;
  answer('%.4f',c^2*x);
@2,0; x,c;
A random variable $X$ has variance $%.2f.$
What is the variance of the random variable $%.1fX?$
>>

<<text;
\bigskip
\noindent {\bf Bernoulli trials.}
>>
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Bernoulli trials
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
<<problemI;
  n={7:11};
  k={2:5};
  p=nchoosek(n,k)/2^n;
  answer('$%s$',sLaTeXFrac(p));
```

```
@3,0; n,k;
Assume that a fair coin is thrown %d times, so that the chance to get
a head or a tail in each throwing is 0.5. What is the probability that
one gets %d heads?
>>


<<problemR;
  n={7:11};
  k={2:5};
  prvl={9:28}/100;
  p=nchoosek(n,k)*prvl^k*(1-prvl)^(n-k);
  answer('%.5f',p);
@3,0; prvl,k,n;
Prevalence of a medical condition in a population is equal to %.2f.
What is the probability that exactly %d people from a random sample
of %d people chosen from this population have the condition?
>>

<<problemR;
  n={7:11};
  k={2:5};
  prvl={21:50}/100;
  p=nchoosek(n,k)*prvl^k*(1-prvl)^(n-k);
  answer('%.5f',p);
@3,0; prvl,k,n;
The probability that a patient with a disease A dies during the next
year is %.2f. What is the probability that exactly %d patients with this
disease from a random sample of %d patients will die during the next year?
>>


<<text;
\bigskip
\noindent {\bf Properties of $\Pr(A).$}
>>

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Properties of Pr(A)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
<<problemT;
%                   False
$\Pr(A+B) = \Pr(A) + \Pr(B).$
If the events $A$ and $B$ are independent then $\Pr(A+B) = \Pr(A)+\Pr(B).$
$\Pr(AB) = \Pr(A)\Pr(B).$
$\Pr(A\mid B) = \Pr(B\mid A).$
If the events $A$ and $B$ are independent then $\Pr(A\mid B) = \Pr(A)\Pr(B).$
The event $A$ is independent form itself.
-----------------------------------------------------------------------
%                   True
$0 \leq \Pr(A) \leq 1.$
$\Pr(A+B) = \Pr(A) + \Pr(B) - \Pr(AB).$
If the events $A$ and $B$ are independent then $\Pr(AB) = \Pr(A)\Pr(B).$
If the events $A$ and $B$ are independent then $\Pr(A\mid B) = \Pr(A).$
$\Pr(AB) = \Pr(A\mid B)\Pr(B).$
$\Pr(A\mid B) = \Pr(B\mid A)\Pr(A)/\Pr(B).$
$\Pr(B) = \Pr(B\mid A) \Pr(A) + \Pr(B\mid \bar A) \Pr(\bar A).$
%
@1,0;6;
%
Let $A$ and $B$ be two random events. Recall that $A+B$ also
denoted as $A \cup B$ is the event that at least $A$ or $B$
```

```
will happen and that $AB$ also denoted as $A \cap B$ is the event
that both $A$ and $B$ will happen. Let also $\bar A$
be the complement of $A.$ How many of the following
assertions are always correct?
>>

<<text;
\bigskip
\noindent {\bf Properties of $E(X)$ and $\Var(X)$}
>>

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Properties of E(X) and Var(X)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
<<problemT;
% False
$x_1+\ldots+x_n=1.$
$E(XY) = E(X)E(Y).$
$E(X^2) = [E(X)]^2.$
$E(\abs{X}) = \abs{E(X)}.$
$E(g(X)) = g(E(X)).$
For any real number $c$ \ $\Var(cX) = c\Var(X).$
If $c$ is a constant then $E(c) = 0.$
If $c$ is a constant then $\Var(c) = c.$
$E(X) \geq 0.$
-----------------------------------------------------------------
% True
$p_1+\ldots+p_n=1.$
$E(X+Y) = E(X)+E(Y).$
For any number $c$ \ $E(cX) = cE(X).$
If $c$ is a constant then $E(c) = c.$
$E(X) = p_1x_1+\ldots+p_nx_n.$
$E(X^2) = p_1x_1^2+\ldots+p_nx_n^2.$
$E(g(X)) = p_1g(x_1)+\ldots+p_n g(x_n).$
$\Var(cX) = c^2\Var(X).$
If $X$ and $Y$ are independent then $E(XY) = E(X)E(Y).$
If $X$ and $Y$ are independent then $\Var(X+Y) = \Var(X) + \Var(Y).$
For any real number $c$ \ $\Var(cX) = c^2\Var(X).$
$\Var(X) = E(X^2)- [(E(X)]^2.$
$\Var(X) \geq 0.$
$E(X^2) \geq 0.$
If $c$ is a constant then $\Var(c) = 0.$
If $\Var(X) = 0$ then $X$ is a constant random variable.
% -----------------------------------------------------------------
@1,0;8;
Let $X$ be a random variable which takes values $x_1,\ldots,x_n$ and only
these values with respective probabilities $p_1,\ldots,p_n,$ let $Y$ be
another random variable and let $g(x)$ be a function. How many of the
following assertions are always correct?
>>

<<text;
\bigskip
\noindent {\bf The Central Limit Theorem}
>>
<<problemI;
  n={101:300};
  k=floor(7*n/15) + randi(round(n/15));
  z=(2*k/n-1)/sqrt(n);
  answer('%.4f',normcdf(z,0,1));
@2,0; n,k;
A fair coin is thrown %d times.
```

```
Find the probability that the number of tails will not exceed %d.
>>


<<problemR;
  n={101:300};
  k=floor(7*n/15) + randi(round(n/15));
  z=(2*k/n-1)/sqrt(n);
  answer('%.4f',normcdf(z,0,1));
@2,0; n,k;
A fair coin is thrown %d times.
Find the probability that the number of tails will not exceed %d.
>>

<<problemR;
  n={101:300};
  p={15:34}/100;
  q=1-p;
  sigma=sqrt(p*q);
  sqrtn=sqrt(n);
  x = n*p + (rand-1/2)*sqrtn*sigma;
  k=round(x);
  z=(k/n-p)*sqrtn/sigma;
  answer('%.4f',normcdf(z,0,1));
@2,0; p,k,n;
Prevalence of a medical condition in a population is equal to %.2f.
What is the probability that no more than %d people from a random sample
of %d people chosen from this population have the condition?
>>

<<problemR;
  n={101:300};
  p=GetRandProbDist(3,1);
  X=randi(10,1,3);
  X=sort(X);
  X=X+[0,1,2];
  EX=sum(p.*X);
  sigma=sqrt(sum(p.*X.*X)-EX^2);

  t = EX+(2*rand-1)*sigma/sqrt(n);
  t=round(t*100)/100;
  z=(t-EX)*sqrt(n)/sigma;
  answer('%0.4f',normcdf(z,0,1));
@2,0; X(1),X(2),X(3),p(1),p(2),p(3),n,n,n,t;
A random variable $X$ takes values $%d,%d$
and $%d$ with probabilities $%.1f,%.1f$ and $%.1f$
respectively. Let $X_1,\ldots,X_{%d}$ be a sequence of mutually
independent random variables which have the same distribution as~$X.$
Let $\bar X = (X_1+\ldots+X_{%d})/{%d}.$ Find the probability
that $\bar X \leq %.3f.$
>>

<<text;
\bigskip
\noindent {\bf Bayes Formula}
>>

<<problemI;
  sns={70:99};
  sns=sns/100;
  spc={70:99};
  spc=spc/100;
```

```
  prvl=(2+randi(20))/100;
  PB=sns*prvl + (1-spc)*(1-prvl);
  answer('%.4f',PB);
@2,0; sns,spc,prvl;
Recall that sensitivity of a disease test is the probability of
a positive test result (a patient has the disease) if he/she indeed has it,
and specificity of a test is the probability of a negative test result
(a patient does not have the disease) if he/she indeed does not have it.
A test for diagnosing a disease has sensitivity %.2f
and specificity %.2f. Prevalence of the disease in a population is %.2f.
Find the probability that for a randomly chosen person from the
population the test would give a positive result.
>>

<<problemI;
  sns={70:99};
  sns=sns/100;
  spc={70:99};
  spc=spc/100;
  prvl=(2+randi(20))/100;
  PB=sns*prvl + (1-spc)*(1-prvl);
  ans = sns*prvl/PB;
  answer('%.4f',ans);
@2,1; sns,spc,prvl;
Recall that sensitivity of a disease test is the probability of
a positive test result (a patient has the disease) if he/she indeed has it,
and specificity of a test is the probability of a negative test result
(a patient does not have the disease) if he/she indeed does not have it.
A test for diagnosing a disease has sensitivity %.2f
and specificity %.2f. Prevalence of the disease in a population is %.2f.
The disease test for a person from that population gave a positive result.
Find the probability that the person actually has the disease.
>>

<<text;
\bigskip
\noindent {\bf Properties of covariance and correlation}
>>

<<problemT;
% False statements
If $\rho(X,Y) = 0$ then the random variables $X$ and $Y$ are independent.
If $\Cov(X,Y) = 0$ then the random variables $X$ and $Y$ are independent.
$-1 \leq \Cov(X,Y) \leq 1.$
For any random variable $X$ \ \ $\rho(X,X) = 0.$
For any random variable $X$ \ \ $\Cov(X,X) = 0.$
-------------------------------------------------------------------------
% True statements
$-1 \leq \rho(X,Y) \leq 1.$
For any random variable $X$ \ \ $\rho(X,X) = 1.$
$\Cov(X,Y) = E[(X-EX)(Y-EY)].$
$\rho(X,Y) = \Cov(X,Y)/\sqrt{\Var(X)\Var(Y)}.$
$\abs{\Cov(X,Y)} \leq \sqrt{\Var(X)\Var(Y)}.$
If $X$ and $Y$ are independent then $\rho(X,Y) = 0.$
If $X$ and $Y$ are independent then $\Cov(X,Y) = 0.$
$\Cov(X,X) = \Var(X).$
If $X$ and $Y$ have normal distributions and $\rho(X,Y) = 0,$ then $X$ and $Y$ are independent.
For any numbers $a$ and $b$ \ \ $\Cov(aX_1+bX_2,Y) = a\Cov(X_1,Y) + b\Cov(X_2,Y).$
$\Cov(X,Y) = \Cov(Y,X).$
@2,1;8;
How many of the following assertions are correct?
>>
```

```
<<text;
\bigskip
\noindent {\bf Confidence intervals.}
>>

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Confidence intervals
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

<<problemI;
  percent='%';
  n={201:500};
  k={101:190};
  phat=k/n;
  sigmahat=sqrt(phat*(1-phat)/n);
  lo=phat-norminv(0.975,0,1)*sigmahat;
  up=phat+norminv(0.975,0,1)*sigmahat;
  answer('$(%.3f,%.3f)$',lo,up);
@3,0; k,n,percent;
A biostatistician would like to estimate prevalence $p$ of a medical
condition in a population. In a trial exactly %d people from a random
sample  of %d people chosen from this population were found to have
the condition. Find 95\%c confidence interval for the prevalence~$p.$
>>

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Poisson distribution
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

<<problemZ;
  n = {1501:5000};
  lmb=n/365;
  k=round(lmb) + 4 - randi(7);
  p=exp(-lmd)*lmb^k/factorial(k);
  answer('$.4f$',p);
@2,0; n,k;
An emergency department of a hospital gets on average %d patients
in a year (365 days). What is the probability that in one day the
emergency department will get %d patients?
>>
```

7. Appendix B: Spikĕ code

```
% Spike, version L3.
% Author: Nurulla Azamov, Flinders University 2014.
% Last modified: 13/11/2014.
%
% Usage: spike(TopicCode,AssNumber).
%
% Example: spike('MATH3701',2).
%
% Input:
%    TopicCode, an 8-symbol topic code.
%    AssNumber, assignment number.
%
%    To run spike, there must be four files:
%      (a) Topfile,
%          this is a file the content of which spike copies
%          into the beginning of the output.
%          Topfile may also contain some Matlab commands to be executed
%          before the spike-code.
%      (b) TopicCodeAn.spk,
%          this is a file with spike-code.
%      (c) TopicCode.txt,
%          this is a file with a list of student ID's
%          enrolled in the topic. The file TopicCode.tex is
%          kept in the topic folder TopicCode.
%          This file may look as follows:
%
%          2071253
%          2022880
%          2095220
%          2119922
%          2107080
%          -------
%
% Output:
%   a LaTeX file with an assignment.
%   This output-LaTeX file is saved in the topic folder.
%   The saved output file is opened in Matlab's editor.


function spike(TopicCode,AssNumber)

fprintf('spike, version L3. Nurulla Azamov, Flinders University 2014.\n');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Initialization of certain parameters.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
MAX_NUMBER_OF_QUESTIONS=80;

% Proportion of total assessment, which this assignment constitutes.
% By default, it is 10%
evalin('base','PercentMarks=10;');

% Assignment due date.
% By default, it is 'to be announced'.
evalin('base','duedate=''TBA'';');

% How many times to repeat each test question.
% By default, only once.
evalin('base','NRepeat=1;');
```

```
% Number of alternative multiple choice answers.
% By default, it is 5.
evalin('base','NAltAns=5;');

% Random seed.
evalin('base','rseed=247;');

% currdir=cd;
% cd('\\userab\a\azam0001\DriveH\Documents\MATLAB\Spike');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Initialization of topic data.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Topic=topic(TopicCode);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Creating the Out-file and opening it for writing.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
OutFile=sprintf('%s\\%s',Topic.code,Topic.code);
if isnumeric(AssNumber),
    OutFile=sprintf('%sA%d.tex',OutFile,AssNumber);
else
    OutFile=sprintf('%sExamPaper.tex',OutFile);
end

OutF=fopen(OutFile,'w'); % the LaTeX out file
if OutF~=-1,
    fprintf('The file %s already exists.\nIt is overwritten.\n',OutFile);
end
fprintf('Please wait ...\n');
fprintf(OutF,'%% This file was created on %s by spike, version L3.\n',date);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Processing the Matlab part of the top-file
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
af=fopen('topfile.tex');

% Process MATLAB commands given in the top-file,
% until the line \documentclass.
% These commands may for instance change some constants.
while ~feof(af),
    ss=fgetl(af);
    if length(ss)>=14 && strcmp(ss(1:14),'\documentclass'),
        fprintf(OutF,'%s\n',ss);
        break
    else
        if (~isempty(ss)) && (ss(1) ~='%'),
            try
                evalin('base',ss);
            catch ME
                fprintf('\n\nSpike: ERROR in top-file.');
                fprintf('\n%s\n\n',ss);
                fprintf('%s\n',ME.identifier);
                fprintf('%s\n',ME.message);
                return
            end
        end
    end
end

if evalin('base','NAltAns > 8'),
    error('The value of NAltAns should not exceed 8.');
```

```
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Initialization of assignment data.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Asst=assignment(Topic,AssNumber);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Processing the LaTeX part of the top-file.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

while ~feof(af),
    ss=fgetl(af);
    if length(ss)>=4 && strcmp(ss(1:4),'%EOF'),
        break
    elseif length(ss)>=16 && strcmp(ss(1:16),'\begin{document}'),
        fprintf(OutF,'\\newcommand{\\topiccode}{%s}\n',Topic.code);
        fprintf(OutF,'\\newcommand{\\topicname}{%s}\n',Topic.name);
        if isnumeric(Asst.number),
            fprintf(OutF,'\\newcommand{\\NAss}{%d}\n',Asst.number);
        else
            fprintf(OutF,'\\newcommand{\\NAss}{Exam Paper}\n');
        end
        fprintf(OutF,'\\newcommand{\\Year}{%d}\n',Asst.year);
        fprintf(OutF,'\\newcommand{\\Semester}{%d}\n',Asst.semester);
        fprintf(OutF,'\\newcommand{\\DueDate}{%s}\n',Asst.duedate);
        fprintf(OutF,'\\newcommand{\\PercentMarks}');
        fprintf(OutF,'{%d}\n',Asst.pmark);
    end

    fprintf(OutF,'%s\n',deblank(ss));
end
fclose(af);
clear ans af ss;
% End of processing of top-file

fprintf(OutF,'%% End of top file.\n');



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Processing the spike-file
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
AnsString =char(ones(Topic.NStud,MAX_NUMBER_OF_QUESTIONS)*32);
Numbers   =char(ones(1,MAX_NUMBER_OF_QUESTIONS)*32);
Weights   =char(ones(1,MAX_NUMBER_OF_QUESTIONS)*32);
Neg_wts   =char(ones(1,MAX_NUMBER_OF_QUESTIONS)*32);
ProblTypes=char(ones(1,MAX_NUMBER_OF_QUESTIONS)*32);
NAltAnswers=char(ones(1,MAX_NUMBER_OF_QUESTIONS)*32);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Start the main cycle
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fprintf(OutF,'\\catcode`@=11\n');

% This is a cycle of assignments.

for n_00 = 1:Topic.NStud,
    fprintf('Preparing variant %d: ',n_00);
    % Prepare an assignment for nth student.
```

```
NSpkLn=0;
spkfile=fopen(Asst.SpkFile);
if spkfile==-1,
    fprintf('Spike: Error.\n');
    fprintf('The file %s is not found.\n',Asst.SpkFile);
    return
end
CurrProbl=0;

fprintf(OutF,'\n\\newpage\n');
fprintf(OutF,'\\renewcommand{\\@oddhead}{{\\small ');
if isnumeric(Asst.number),
    fprintf(OutF,' %s Assignment %d \\ ',Topic.code,Asst.number);
    fprintf(OutF,' Variant %d \\ %%\n',n_00);
    fprintf(OutF,' %d \\ Semester %d',Asst.year,Asst.semester);
else
    fprintf(OutF,' %s Exam Paper %d',Topic.code,n_00);
end
fprintf(OutF,'\\quad \\hfil {\\sf Student ID: ****');
ID_LEN=Topic.STUDENT_ID_LENGTH;
fprintf(OutF,'%s}}}\n',Topic.StudentList(n_00,ID_LEN-2:ID_LEN));
fprintf(OutF,'\\setcounter{nques}{0}\n\n');

% This is a cycle of problems in an assignment.
while true,
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%% Read one spike-block.
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Fields of a spike-block:
    % Cmd: command,
    % NCmdPt: number of commands in the command part,
    % CmdPt: the command part of a spike-block,
    % NTxtPt: number of lines in the text part,
    % TxtPt: the text part of a spike-block,
    % AtPar: parameters of @-line,
    % TxtPrPar: parameters of the text part,
    % SpkLn: number of the current line in spike-file.
    [SB,status,NSpkLn]=ReadSpikeBlock(spkfile,NSpkLn);

    if strcmp(status,'error'),
        return
    elseif strcmp(status,'EOF') || feof(spkfile),
        % if end of spike file, then ...
        if n_00==1,
            Number_of_problems=CurrProbl;
        end
        break
    end

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % A spike-block <<...>> has been read.
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Processing of the spike-block.
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%  text  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    if strcmp(SB.Cmd,'text'),
        for k_00=1:SB.NTxtPt,
```

```
            fprintf(OutF,'%s\n',deblank(SB.TxtPt(k_00,:)));
        end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%  data  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    elseif strcmp(SB.Cmd,'data'),
        rng(Asst.rcode+Asst.rseed*n_00);
        Q=datablock(SB);
        Q.ProcessCmdPt;
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%  problem Z (void) %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    elseif strcmp(SB.Cmd,'problemZ'),
        % in this case, do nothing
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%  problems I,R,T,G,H,V  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    else
        Q=eval(sprintf('%s(SB,OutF)',SB.Cmd)); % Constructor
        NR=evalin('base','NRepeat');
        for rep_00=1:NR,
            fprintf(OutF,'\n\\problemtype{%c}\n',Q.Cmd(8));
            CurrProbl=CurrProbl+1;
            fprintf('%d',mod(CurrProbl,10)); % echo
            rng(Asst.rcode+Asst.rseed*n_00+CurrProbl);
            Q=Q.Execute;
            AnsString(n_00,CurrProbl)=Q.CorrAns;

            if n_00==1,
                Numbers(CurrProbl)=num2str(mod(CurrProbl,10));
                Weights(CurrProbl)=char(Q.Marks(1)+'0');
                Neg_wts(CurrProbl)=char(Q.Marks(2)+'0');
                ProblTypes(CurrProbl)=Q.Cmd(8);
                k=evalin('base','NAltAns');
                NAltAnswers(CurrProbl) = char(k+'0');
            end
        end
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%  End of spike block <<...>>  processing %%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  end
  fprintf(OutF,'%%================================================\n');
  fprintf(OutF,'%%================================================\n');
  fclose(spkfile);
  fprintf('  done.\n');
end

fprintf(OutF,'\n\\catcode`@=12\n\\end{document}\n');


fprintf(OutF,'\n%%================================================');
fprintf(OutF,'\n%%============= Marking information ================');
fprintf(OutF,'\n%%================================================');

fprintf(OutF,'\n\nRandom seed: %X\n',Asst.rseed');

fprintf(OutF,'\n\nThe following information is given ');
fprintf(OutF,'for marking purposes.\nThe line which ');
fprintf(OutF,'starts with %%MARKING indicates beginning of ');
fprintf(OutF,'the marking\ninformation. The marking program ');
fprintf(OutF,'looks for this line and once it is found\nthe ');
```

```
fprintf(OutF,'marking program starts the marking process.\nBefore the');
fprintf(OutF,' seven dash line ------- correct answers are given.\n');
fprintf(OutF,'After ------- students answers should be given.\n\n');

fprintf(OutF,' PrbType: types of questions: A,B,C,F or G (in version L3).\n');
fprintf(OutF,' Numbers: the last digit of the number of question.\n');
fprintf(OutF,' Weights: mark for a correct answer.\n');
fprintf(OutF,' Neg wts: penalty for an incorrect answer.\n');
fprintf(OutF,' Specfcs: specifications for marking:\n');
fprintf(OutF,'  (a) "+" consider an answer as correct (for whatever reason).\n');
fprintf(OutF,'  (b) "0" ignore the question, that is, give no marks for it.\n');
fprintf(OutF,'  (c) "n" do not show the right answer for this question.\n');
fprintf(OutF,'  (d) "." or " " treat the question as usual.\n\n');

fprintf(OutF, 'Topic code:          %s\n',Topic.code);
if isnumeric(Asst.number),
    fprintf(OutF, 'Assignment number:   %d\n',Asst.number);
else
    fprintf(OutF, 'Exam paper.\n');
end
fprintf(OutF, 'Number of questions: %d\n\n',Number_of_problems);

fprintf(OutF, 'PrbType:%s\n',ProblTypes);
fprintf(OutF, 'NAltAns:%s\n',NAltAnswers);
fprintf(OutF, '%%MARKING\n');
fprintf(OutF, 'Numbers:%s\n',Numbers);
fprintf(OutF, 'Specfcs:%s\n',char(ones(1,Number_of_problems)*'.'));
fprintf(OutF, 'Weights:%s\n',Weights);
fprintf(OutF, 'Neg wts:%s\n',Neg_wts);
for n=1:Topic.NStud,
    fprintf(OutF, '%s:%s\n',Topic.StudentList(n,:),AnsString(n,:));
end
fprintf(OutF, '-------\n');
fprintf(OutF, '%%Insert here students'' answers.\n');
fclose(OutF);

% cd(currdir);
fprintf('Done.\n\n');

edit(OutFile);

end   % spike
```

```
classdef assignment

    properties
        topic;
        number;
        pmark;    % mark for this assignment in percents.
        year;
        semester;
        duedate;
        rcode;    % random code
        rseed=57; % random seed. These two numbers are used to
                  %    initialize Matlab's generator of random numbers.
        SpkFile;
    end

    methods
        function A=assignment(topic0,number0)
            if (nargin>0),
                A.topic=topic0;
                A.number=number0;
                A.pmark=evalin('base','PercentMarks');
                A.rseed=evalin('base','rseed');
                A.duedate=evalin('base','duedate');

                c=clock; A.year=round(c(1));
                if c(2)<=6,
                    A.semester = 1;
                else
                    A.semester = 2;
                end

                % Random code of the assignment: YEAR + SEMESTER.
                A.rcode=A.year*10+A.semester;
            end % if (nargin>0)

            if isnumeric(A.number),
                A.SpkFile=sprintf('%s\\%sA%d.spk',A.topic.code,A.topic.code,A.number);
            elseif A.number=='E',
                A.SpkFile=sprintf('%s\\%sEx.spk',A.topic.code,A.topic.code);
            else
                fprintf('Spike: Error.\nThe second argument must ');
                fprintf('be a number or ''E'' character.\n');
                return
            end

        end % function assignment
    end % methods
end
```

```
classdef datablock < spikeblock

    methods

        function q=datablock(S)
            q=q@spikeblock(S.SpkLn,S.Cmd,S.CmdPt,S.NCmdPt,S.TxtPt,S.NTxtPt,S.AtPar);
        end

        function q=ProcessCmdPt(q)
            % Evaluate commands of the command part.
            % If an attempt to evaluate fails, give an error message.
            for k=1:q.NCmdPt,
                out = q.CmdPt(k,:);
                try
                    Spike2Matlab(out);
                catch ME
                    q.ME_Error(out,ME);
                end
            end
        end

    end

end
```

```
classdef problem < spikeblock
    % problem is the superclass of problemI, problemR, etc
    %

    properties
        NAltAnswers;   % number of alternative answers A,B,C,...
        outfile;   % the output LaTeX file
        CorrAns;   % correct answer, one of these: A,B,C,...
        TxtPtPar;  % parameters
        Marks;     % marks(1) is the mark for correct answer,
                   % -markts(2) is the mark for incorrect answer.
    end

    methods
        function q=problem(S,outfile0)
            q=q@spikeblock(S.SpkLn,S.Cmd,S.CmdPt,S.NCmdPt,S.TxtPt,S.NTxtPt,S.AtPar);
            q.outfile=outfile0;
            [vars,num]= SeparateVariables(q.AtPar,';');
            if num<3,
                fprintf('\nSpike: ERROR in line %d:',q.SpkLn);
                fprintf(' @-line must have at least two semicolons.\n');
                return
            end
            q.AtPar=strtrim(vars(1,:));
            q.TxtPtPar=[',',strtrim(vars(2,:))];
            if num>=4,
               ss=strtrim(vars(3,:));
               q.NAltAnswers= ss(1)-'0';
            else
                q.NAltAnswers=evalin('base','NAltAns');
            end
        end

        function q=Execute(q)
            q=q.ProcessCmdPt;
            q=q.SetRandPerm;
            q=q.SetCorrAns;
            q=q.WriteAtLine;
            q=q.ProcessTxtPt;
            q=q.ProcessAnswers;
        end

        function Sp_Error(q,message,str)
            fprintf('\n\n\nSpike: error in spike-block ');
            fprintf('starting at line %d in spk-file.\n',q.SpkLn);
            fprintf('Spike-block type: %s\n',q.Cmd);
            fprintf('Message:\n      %s\n',message);
            fprintf('Line with error:\n      !%s!\n',str);
            fprintf('Press Ctrl+Pause to terminate the program.\n');
            pause;
        end

    end

    methods (Access = protected)

        function q=SetCorrAns(q)
        end

        function q=SetRandPerm(q)
        end
```

```
        function q=ProcessCmdPt(q)
        end

        function q=WriteAtLine(q)
            [vars,len]=SeparateVariables(q.AtPar,',');
            q.Marks=zeros(1,max(2,len));
            q.Marks(1)=str2double(vars(1,:));
            q.Marks(2)=0;
            for j=2:len,
                q.Marks(j)=str2double(vars(j,:));
            end
            fprintf(q.outfile,'\\probl%c{%c}','A'-1+len,q.CorrAns);
            for j=1:len,
                fprintf(q.outfile,'{%d}',q.Marks(j));
            end
            fprintf(q.outfile,'\n');
        end

        function q=ProcessTxtPt(q)
        end

        function q=ProcessAnswers(q)
        end


    end

end
```

```
classdef problemG < problem

    properties
        nques;   % number of statements to be chosen
        nchosentrue; % number of true statements in a chosen random sample
    end

    methods

        function q=problemG(S,outfile0)
            q=q@problem(S,outfile0);
            % Number of true/false statements which are to be chosen;
            % this number is given at @ line as third number.
            % For example, @2,0;7; means we have to choose 7 statements.
            if ismember(q.TxtPtPar(2:end),'0123456789'),
                q.nques=str2double(q.TxtPtPar(2:end));
            else
                q.Sp_Error('Incorrect format of the @-line',q.TxtPtPar(2:end));
            end
        end

    end

    methods (Access = protected)


        function q=SetCorrAns(q)
           q.CorrAns=char(64+randi(q.NAltAnswers));
        end

        function q = ProcessTxtPt(q)
            % Write the body of the multiple choice question;
            % it is of this kind: ``How many of the following
            % statements are true?''
            for k=1:q.NTxtPt,
                fprintf(q.outfile,'%s\n',deblank(q.TxtPt(k,:)));
            end

            % nchosentrue is the number of right statements chosen randomly.
            q.nchosentrue=0;
            fprintf(q.outfile,'\\begin{enumerate}\n');
            for k=1:q.nques,
                [s,b]=eval(q.CmdPt(1,:));
                fprintf(q.outfile,'\\item %s\n',s);
                q.nchosentrue=q.nchosentrue+b;
            end
            fprintf(q.outfile,'\\end{enumerate}\n\n');
        end


        function q=ProcessAnswers(q)
            % Get multiple choice answers.
            pp=MCQAnswers(q.nchosentrue,q.CorrAns-64,0,q.nques,q.NAltAnswers);

            % Print multiple choice answers.
            for k=1:q.NAltAnswers,
                fprintf(q.outfile,'(%c) \\ %d \\quad \n',char(64+k),pp(k));
            end
        end

    end
```

```
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%   Subfunctions   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function
pp=MCQAnswers(NChosenTrue,CorrAnsPos,LowBound,UpperBound,NAltAns)
% This function returns a string of multiple choice answers.
% Here NChosenTrue is the correct answer, for example, 7;
% CorrAnsPos is the position of this correct answer, for example, 4;
% LowBound and UpperBound are lower and upper bounds for the other four
% alternative wrong answers, for example, LowBound =0 and UpperBound=9.
% Then for NAltAns=5 MCQAnswers will produce something like
% (A) 2  (B) 0  (C) 8  (D) 7  (E) 5

% a is a random set of NAltAns-1 numbers
% chosen from the set LowBound..UpperBound \ NChosenTrue.
a=LowBound-1+randperm(UpperBound-LowBound,NAltAns-1); a = a +
(a>=NChosenTrue);
pp=[a(1:CorrAnsPos-1),NChosenTrue,a(CorrAnsPos:NAltAns-1)]; end
```

```
classdef problemH < problemG

    methods

        function q=problemH(S,outfile0)
            q=q@problemG(S,outfile0);
        end

    end

    methods (Access = protected)


        function q=ProcessCmdPt(q)
            % Execute commands of the command part.
            % If an attempt to execute fails, give an error message.
            for k=1:q.NCmdPt,
                out = q.CmdPt(k,:);
                try
                    Spike2Matlab(out);
                catch ME
                    q.ME_Error(out,ME);
                end
            end
        end

        function q = ProcessTxtPt(q)
            % Write the body of the multiple choice question;
            % it is of this kind: ‘‘How many of the following
            % statements are true?’’
            for k=1:q.NTxtPt,
                fprintf(q.outfile,'%s\n',deblank(q.TxtPt(k,:)));
            end

            % nchosentrue is the number of right statements chosen randomly.
            q.nchosentrue=0;
            fprintf(q.outfile,'\\begin{enumerate}\n');
            for k=1:q.nques,
                q.ProcessCmdPt;
                s=evalin('base','out');
                fprintf(q.outfile,'\\item %s\n',s);
                q.nchosentrue=q.nchosentrue+evalin('base', 'tf');
            end
            fprintf(q.outfile,'\\end{enumerate}\n\n');
        end

    end

end
```

```
classdef problemI < problem

    properties
        RandPerm; % random permutation of answers.
        answer;   % the answer string which CmpPt returns upon execution.
    end

    methods

        function q=problemI(S,outfile0)
            q=q@problem(S,outfile0);
        end

    end

    methods (Access = protected)

        function q=SetRandPerm(q)
            q.RandPerm = randperm(q.NAltAnswers);
        end

        function q=SetCorrAns(q)
            % We choose a random permutation of answers so that
            % RandPerm(CorrAns)=1.
            for k=1:q.NAltAnswers,
                if q.RandPerm(k)==1,
                    q.CorrAns=char(64+k);
                    break
                end
            end
        end

        function q=ProcessCmdPt(q)
            % Execute commands of the command part.
            % If an attempt to evaluate fails, give an error message.
            for k=1:q.NCmdPt-1,
                out = q.CmdPt(k,:);
                try
                    Spike2Matlab(out);
                catch ME
                    q.ME_Error(out,ME);
                end
            end
            ansstr=q.CmdPt(q.NCmdPt,:);
            ansstr=strtrim(ansstr);
            if length(ansstr) < 7 || ~strcmp(ansstr(1:7),'answer('),
                q.Sp_Error('Answer operator is expected',ansstr);
            else
                q.answer=evalin('base',['sprintf',ansstr(7:end)]);
            end
        end

        function q=ProcessTxtPt(q)
            params=q.TxtPtPar;
            for k=1:q.NTxtPt,
                ss=deblank(q.TxtPt(k,:));
                ss=strrep(ss,'\','\\');
                ss=strrep(ss,'''','''''');

                % Get number of parameters.
```

```
        NumOfPercents=sum(ismember(ss, '%'));
        [pars,params]=GetParameters(NumOfPercents, params);
        if strcmp(params,'!ERROR!'),
            msg='Insufficient number of parameters.';
            q.Sp_Error(msg,q.TxtPtPar(2:end));
        end

        % Print the line of the text part to the output
        % LaTeX file, processing the parameters.

        out=['sprintf(''',ss,'''',pars,')'];
        try
            outstr=evalin('base',out);
        catch ME
            q.ME_Error(out,ME);
        end
        fprintf(q.outfile,'%s\n',outstr);
    end
    if ~isempty(params),
        msg='Too many parameters.';
        q.Sp_Error(msg,q.TxtPtPar(2:end));
    end
end

function q=ProcessAnswers(q)
    MAX_ANSWER_LENGTH=147;
    Good_Answer_Attempts=evalin('base','GOOD_ANSWER_ATTEMPTS');
    answ=char(ones(q.NAltAnswers,MAX_ANSWER_LENGTH)*32);
    anslen=zeros(1,q.NAltAnswers);
    nattempts=0;

    anslen(1)=length(q.answer);
    if anslen(1) > MAX_ANSWER_LENGTH,
        q.Sp_Error('answer line is too long',q.answer);
    elseif anslen(1) ==0,
        q.Sp_Error('answer line is empty','');
    end
    answ(1,1:anslen(1))=q.answer;

    for m=2:q.NAltAnswers
        while nattempts<1000,
            q=q.ProcessCmdPt;

            % Check that the current answer is different
            % from the previous ones.
            answer_is_good=true;
            for j=1:m-1,
                if strcmp(q.answer,answ(j,1:anslen(j))),
                    answer_is_good=false;
                    break
                end
            end

            if answer_is_good,
                anslen(m)=length(q.answer);
                if anslen(m) > MAX_ANSWER_LENGTH,
                    q.Sp_Error('answer line is too long',q.answer);
                end
                answ(m,1:anslen(m))=q.answer;
                break
            else
                nattempts=nattempts+1;
```

```
                        if nattempts>=Good_Answer_Attempts,
                            fprintf('\nSpike: too many attepmts to ');
                            fprintf('find a non-repeating answer.\n');
                            fprintf('q.NAltAnswers: %d\n',q.NAltAnswers);
                            fprintf('\nContinue? (Y/N)');
                            yn=input('','s');
                            if (yn=='Y') || (yn=='y'),
                                fprintf('\nSpike: I''ll try 50 more times.\n');
                                Good_Answer_Attempts=Good_Answer_Attempts+50;
                            else
                                fclose(q.outfile);
                                fprintf('The program has been terminated.\n');
                                return
                            end
                        end
                        continue
                    end
                end
            end


            fprintf(q.outfile,'\n\\medskip\\noindent\n');

            % Write Multiple Choice answers.
            for k=1:q.NAltAnswers,
                ik=q.RandPerm(k);
                fprintf(q.outfile,'(%c) \\ ',char(64+k));
                fprintf(q.outfile,'%s',answ(ik,1:anslen(ik)));
                fprintf(q.outfile,' \\quad \n');
            end
            fprintf(q.outfile,'\n');
        end
    end

end

function [ss,s2]=GetParameters(n,params)
  ss='';
  for j=1:n,
      [a,params]=strtok(params,',');
      if isempty(a),
          s2='!ERROR!';
          return
      end
      ss=sprintf('%s,%s',ss,a);
  end
  s2=params;
end
```

```
classdef problemR < problemI

    properties
        lastdigit;
    end

    methods

        function q=problemR(S,outfile0)
            q=q@problemI(S,outfile0);
        end

    end

    methods (Access = protected)


        function q=SetRandPerm(q)
            % the identity permutation
            q.RandPerm = 1:q.NAltAnswers;
        end

        function q=SetCorrAns(q)
            anssize=length(q.answer);
            if ~ismember(q.answer(anssize),'1234567890'),
                q.Sp_Error('Answer string must end by a digit',q.answer);
            end

            q.lastdigit=str2double(q.answer(anssize));
            q.CorrAns=char(floor(q.lastdigit/2) + 65);
        end


        function q=ProcessAnswers(q)
            fprintf(q.outfile,'\n\\medskip\\noindent\n');
            % Write Multiple Choice answers.
            if  mod(q.lastdigit,2)==0,
                lastchar='02468';
            else
                lastchar='13579';
            end
            for m=1:q.NAltAnswers,
                fprintf(q.outfile,'(%c) \\ ',char(64+m));
                fprintf(q.outfile,'$%s',q.answer(1:end-1));
                fprintf(q.outfile,'%c$\\quad\n',lastchar(m));
            end
            fprintf(q.outfile,'\n');
        end
    end

end
```

```
classdef problemT < problemG

    properties
        ntrue;   % number of true statements
        nfalse;  % number of false statements
        p_00;    % random sample of statements
    end

    methods

        function q=problemT(S,outfile0)
            q=q@problemG(S,outfile0);
        end

    end

    methods (Access = protected)

        function q = ProcessCmdPt(q)
            MAX_COMMAND_LENGTH=780;
            for j=1:q.NCmdPt,
                % Process jth line of the command part.
                s_00=ProcessFString(q.CmdPt(j,:));
                q.CmdPt(j,:)=char(ones(1,MAX_COMMAND_LENGTH)*32);
                q.CmdPt(j,1:length(s_00))=s_00;
            end

            % The lines which follow after @ line are for
            % the question body. The @ line itself in type T
            % question must have the format @N1[,N2[,N3]]; k;
            % where k is the number of statements to choose.

            q.ntrue=0;
            for j=1:q.NCmdPt,
                if strcmp(q.CmdPt(j,1:7),'-------'),
                    q.nfalse=j-1;
                    q.ntrue=q.NCmdPt-q.nfalse-1;
                    break
                end
            end
            if q.ntrue==0,
                q.Sp_Error('False and true statements should be seperated by -------',['@',q.AtPar]);
            end

            % Decide which statements to pick up
            q.p_00=randperm(q.nfalse+q.ntrue,q.nques);
            % skip ``seven dash'' (that is, -------) line
            q.p_00 = q.p_00 + (q.p_00>q.nfalse);
            % number of correct statements chosen.
            q.nchosentrue=sum(q.p_00>q.nfalse);
        end

        function q = ProcessTxtPt(q)
            % Print a sentence of this kind:
            % ``How many of the following statements are true?''.
            for k=1:q.NTxtPt,
                fprintf(q.outfile,'%s\n',deblank(q.TxtPt(k,:)));
            end
            fprintf(q.outfile,'\\begin{enumerate}\n');
            for k=1:q.nques,
                fprintf(q.outfile,'\\item ');
```

```
                fprintf(q.outfile,'%s\n',deblank(q.CmdPt(q.p_OO(k),:)));
            end
            fprintf(q.outfile,'\\end{enumerate}\n\n');
        end

    end

end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%   Subfunctions   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function outss=ProcessFString(inss)
% This function is used to process lines in questions of type problemT.

if strncmp(inss,'-------',7),
    outss=inss;
else
    [a,b]=strtok(inss,'@');
    b=strtrim(b);
    if isempty(b),
        outss=a;
        return
    end
    b=ProcessCurvedBrackets(b(2:end));
    if isempty(b),
        outss=a;
    else
        a=strrep(a,'\','\\');
        a=strrep(a,'''','''''');
        str0=['sprintf(''',a,''',',b,');'];
        try
            outss=eval(str0);
        catch ME
            fprintf('\nSpike: ERROR in line ''%s'':\n',inss);
            fprintf('%s\n',ME.identifier);
            fprintf('%s\n',ME.message);
            error('The program is halted due to an error.');
        end
    end
end
end
```

```
classdef problemV < problem

    properties
    end

    methods

        function q=problemV(S,outfile0)
            q=q@problem(S,outfile0);
        end

    end

    methods (Access = protected)

        function q=SetCorrAns(q)
            q.CorrAns=q.TxtPtPar(2);
        end

        function q=ProcessTxtPt(q)
%             q.AtPar=deblank(q.AtPar);
%             if (length(q.AtPar)<5) || (isempty(intersect(q.AtPar(5),'ABCDEFGH'))),
%                 q.Sp_Error('wrong or missing answer',['@',q.AtPar]);
%             end
            for k=1:q.NTxtPt,
                fprintf(q.outfile,'%s\n',deblank(q.TxtPt(k,:)));
            end
            fprintf(q.outfile,'\n');
        end

    end
end
```

```
function outss=ProcessCurvedBrackets(inss)

  % if the string inss contains the substring 'sprintf'
  % then do nothing.
  s=strfind(inss,'sprintf');
  if ~isempty(s),
      outss=inss;
      return
  end

  % if the string inss does not contain curved brackets
  % then do nothing.
  [a,b]=GetPairOfCurvedBrackets(inss);
  if a==0,
      outss=inss;
      return
  end

  pp=evalin('base',sprintf('[%s]',inss(a+1:b-1)));
  len=length(pp);
  nn=pp(randi(len));
  outss=sprintf('%s%d%s',inss(1:a-1),nn,inss(b+1:end));
  outss=ProcessCurvedBrackets(outss);
end

function [a,b]=GetPairOfCurvedBrackets(inss)
% a and b are indices of the first inner pair of brackets { and }.
% For example, if inss='a={3:{4:10}}', then a = 6 and b = 11.
a=0; b=0; n=length(inss); for j=1:n,
    if inss(j)=='{',
        a=j;
    elseif inss(j)=='}',
        if a==0,
            error('Unbalanced pair of curved brackets {...}.');
        else
            b=j;
            return
        end
    end
end
end
```

```
function [SB,status,SpkLn]=ReadSpikeBlock(spkfile,SpkLn)
% This function reads one spike-block <<...>> from a text file spkfile,
% and returns it in the spike-block SB.
% Last edited on 27/11/2014.
%
%
% Input: a pointer is at line SpkLn outside a spike-block of a file spkfile.
%
% Output: one spike-block <<...>> is read into SB and the pointer is at
% last line of this spike-block.
%


MAX_NUMBER_OF_COMMANDS=500;
MAX_COMMAND_LENGTH=780;
MAX_NUMBER_OF_PROBLTXT=87;
MAX_PROBLTXT_LENGTH=237;

Cmd='EOF';
CmdPt='';
NCmdPt=0;
TxtPt='';
NTxtPt=0;
AtPar='';
status='';
SB='';

% Read an opening command << and the Cmd name.
while true,
    if feof(spkfile),
        status='EOF';
        return
    else
        s=fgetl(spkfile);
        s=deblank(s);
        SpkLn=SpkLn+1;
        if (length(s)>=2) && (strcmp(s(1:2),'<<')),
            SpkLn0=SpkLn;
            if s(end)~=';',
                fprintf('\nSpike: error in line %d: ; is expected at the end of line.\n',SpkLn);
                status='error';
                return
            else
                Cmd=s(3:end-1);
                break
            end
        elseif (length(s)>=4) && (strcmp(s(1:4),'%EOF')),
            status='EOF';
            return
        end
    end
end

if ~strcmp(Cmd,'data') && ~strcmp(Cmd,'text'),
    if ~strcmp(Cmd(1:7),'problem') || isempty(intersect(Cmd(8),'GHIRVTZ')),
        fprintf('\nSpike: error in line %d: after << one of the following commands is expected:\n',SpkLn);
        fprintf('data, text, problem[G,H,I,R,V,T,Z].\n');
        status='error';
        return
    end
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Read the command part CmdPt of spike-block.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The command part CmdPt is what comes between << and @.

% Here we are inside << ... >>

% If the spike block is not a text-block, then read
% the command part of the spike block.
if ~strcmp(Cmd,'text'),

    % Initialize CmdPt as an array of MAX_NUMBER_OF_COMMANDS empty commands.
    % 32 is the code of empty space character.
    CmdPt=char(ones(MAX_NUMBER_OF_COMMANDS,MAX_COMMAND_LENGTH)*32);

    while true,
        if feof(spkfile),
            fprintf('\nSpike: error in line %d: @ or >> is expected.\n',SpkLn);
            status='error';
            return;
        else
            s=fgetl(spkfile);
            s=deblank(s);
            SpkLn=SpkLn+1;
            if (length(s)>=4) && (strcmp(s(1:4),'%EOF')),
                fprintf('\nSpike: error in line %d: @ or >> is expected.\n',SpkLn);
                status='error';
                return
            end
        end

        % Here s is a current non-empty line from a spike-file.
        if ~isempty(s) && s(1)=='%',

            continue

        elseif ~isempty(s) && strcmp(s(1),'@'),

            if strcmp(Cmd,'data'),
                fprintf('\nSpike: error in line %d: >> was expected, @ was found.\n',SpkLn);
                fprintf('\nData spike-block should not have @...>> part.\n');
                status='error';
                return
            else
                break % from << ... @
            end

        elseif length(s)>=2 && strcmp(s(1:2),'>>'),

            break % from << ... >>

        else % read the command lines between << and @
            len=length(s);
            while ~isempty(s) && s(len)=='%',
                if feof(spkfile),
                    fprintf('\nSpike: error in line %d: @ or >> is expected.\n',SpkLn);
                    status='error';
                    return
                end
                t=fgetl(spkfile);
                SpkLn=SpkLn+1;
```

```
                s=[s(1:len-1),t];
                len=length(s);
            end
            if length(s)>MAX_COMMAND_LENGTH,
                fprintf('\nSpike: a command length is larger than %d.',MAX_COMMAND_LENGTH);
                status='error';
                return
            end
            NCmdPt=NCmdPt+1;
            if ~isempty(s),
                CmdPt(NCmdPt,1:length(s))=s;
            end
        end
    end

    CmdPt=CmdPt(1:NCmdPt,:);

end % if ~strcmp(Cmd,'text')

if ~isempty(s) && strcmp(s(1),'@'),
    AtPar=s(2:end);
end

% At this stage we are inside a subblock @...>>
% or inside a text-block <<...>>.
% Read the text part of <<...>> to the array TxtPt.
% The text part is what comes between @ and >>

if ~strcmp(Cmd,'data'),

    TxtPt=char(ones(MAX_NUMBER_OF_PROBLTXT,MAX_PROBLTXT_LENGTH)*32);

    while true,
        if feof(spkfile),
            error('Spike: >> is expected');
        end

        % read a line, ignoring comments.
        s=fgetl(spkfile);
        s=deblank(s);
        SpkLn=SpkLn+1;
        if ~isempty(s) && s(1)=='%',
            continue
        end

        if (length(s)>=4) && (strcmp(s(1:4),'%EOF')),
            fprintf('\nSpike: error in line %d: @ or >> is expected.\n',SpkLn);
            status='error';
            return
        end

        if (length(s)>=2) && (strcmp(s(1:2),'>>')),
            % if this is the end of the problem part, ...
            break
        else
            NTxtPt=NTxtPt+1;
            if ~isempty(s),
                TxtPt(NTxtPt,1:length(s))=s;
            end
        end
    end
```

```
    TxtPt=TxtPt(1:NTxtPt,:);

end % if ~strcmp(Cmd,'data') && ...

SB=spikeblock(SpkLn0,Cmd,CmdPt,NCmdPt,TxtPt,NTxtPt,AtPar);

end
```

```
function [vars,len]=SeparateVariables(ss,sep)
% For example, if sep==',' and ss=='a1,xy,k'
% this function will return len=3 and vars(1)='a1',
% vars(2) = 'xy' and vars(3)='k '.
%

ss(ss==' ')='';
seppos=[0,strfind(ss,sep),length(ss)+1];
len=sum(ss==sep)+1;
vars=char(ones(len,20)*32);
for j=1:len,
    vars(j,1:seppos(j+1)-seppos(j)-1)=ss(seppos(j)+1:seppos(j+1)-1);
end

end
```

```
function Spike2Matlab(inss)
% Generally, a Spike command is just a Matlab command, but with
% one exception. This function transforms that exceptional
% Spike command into Matlab format.
%
% Specifically,
%    if a spike-command sb has a pair of curved brackets {...}
%    then this function replaces them by a number,
%    chosen randomly from the list of numbers inside the curved brackets.
%
%    For example, a command such as
%
%       a=[{3:7},{3,5,7,11}];
%
%    will be replaced in one run by, say, a=[6,7],
%    in another run by, say, a=[4,11], etc.
%
% Input: sb, a string with a Spike operator;
% Output: sa, a string with a MATLAB operator.

  if MultipleAssign(inss),
      outss=ProcessCurvedBrackets(inss);
      evalin('base',outss);
  end
end



function bool=MultipleAssign(ss)

bool=0;
inss=strtrim(ss);
order = 0;
k=strfind(inss,'!=');
if isempty(k),
    k=strfind(inss,'!<');
    if isempty(k),
        bool=1;
        return
    else
        order = 1;
    end
end

if length(k)>1,
    error(['Incorrect multiple assignment operator(1): ',inss]);
end
if ~strcmp(inss(end-1:end),'};') || ~strcmp(inss(k+2),'{'),
    error(['Incorrect multiple assignment operator(2): ',inss]);
end

pp=evalin('base',sprintf('[%s]',inss(k+3:end-2)));

[vars,lenvar]=SeparateVariables(inss(1:k-1),',');
if lenvar==1,
    error('At least two variables are expected.');
end
ind=randperm(length(pp),lenvar);
a=pp(ind);
if order,
    a = sort(a);
```

```
end

if isnumeric(a(1)),
    for j=1:lenvar,
        evalin('base',sprintf('%s=%d;',deblank(vars(j,:)),a(j)));
    end
else
    for j=1:lenvar,
        evalin('base',sprintf('%s=''%c'';',deblank(vars(j,:)),a(j)));
    end
end

end
```

```
classdef spikeblock

    properties
        SpkLn;     % the number of the line in a spike-file
                   %         where this block starts.
        Cmd;       % one of these: data, text, problemR, problemI,
                   %               problemT, problemV or problemG.
        CmdPt;     % the command part of spike-code;
                   %         this is what comes between << and @
        NCmdPt;    % the number of lines in the command part;

        TxtPt;     % the text part of spike-code;
                   %         this is what comes between @ and >>
        NTxtPt;    % the number of lines in the text part;

        AtPar;     % This is the line in the block
                   %         which starts with @ character;
    end

    methods
        function q=spikeblock(spkLn,cmd,cmdPt,nCmdPt,txtPt,nTxtPt,atPar)
            q.SpkLn=spkLn;
            q.Cmd=cmd;
            q.CmdPt=cmdPt;
            q.NCmdPt=nCmdPt;
            q.TxtPt=txtPt;
            q.NTxtPt=nTxtPt;
            q.AtPar=atPar;
        end

        function ME_Error(q,str,ME)
            fprintf('\n\n\nSpike: error in spike-block ');
            fprintf('starting at line %d in spk-file.\n',q.SpkLn);
            fprintf('Spike-block type: %s\n',q.Cmd);
            fprintf('Line with error:\n      !%s!\n',str);
            fprintf('%s\n',ME.identifier);
            fprintf('%s\n',ME.message);
            fprintf('Press Ctrl+Pause to terminate the program.\n');
            pause;
        end
    end
end
```

```
classdef topic

    properties(Constant=true)
        MAX_NUMBER_OF_STUDENTS=500;
        STUDENT_ID_LENGTH=7;
    end

    properties
        code;
        name;
        NStud;            % number of students enrolled in this topic.
        studentsfilename; % file with list of students enrolled in this topic.
        sfile;            % file variable for studentsfilename.
        StudentList;
    end

    methods
        function t=topic(topiccode)
            topiccode=upper(topiccode);
            t.code=topiccode;

            if     strcmp(topiccode,'MATH2712'),
                t.name='Algebra';
            elseif strcmp(topiccode,'MATH3712'),
                t.name='Partial Differential Equations';
            elseif strcmp(topiccode,'MATH3701'),
                t.name='Numerical Analysis';
            elseif strcmp(topiccode,'MATH4709'),
                t.name='Topology';
            else
                t.name='Unknown Topic Code';
            end

            t.studentsfilename=sprintf('%s\\%s.txt',t.code,t.code);

            t.sfile=fopen(t.studentsfilename);
            if t.sfile==-1,
                error('Spike: student file is not found.');
            end
            % NStud is the number of students.
            t.NStud=0;
            % 48 is the code of the character ``0''
            t.StudentList=char(ones(t.MAX_NUMBER_OF_STUDENTS,t.STUDENT_ID_LENGTH)*48);

            % Read the student ID's from the student
            % ID file to the array StudentList.
            while ~feof(t.sfile),
                ss=fgetl(t.sfile); % ID of a student
                if length(ss) >= 1 && ss(1)=='%',
                    continue
                elseif length(ss)<t.STUDENT_ID_LENGTH,
                    fprintf('Spike: error in a student list file:\n');
                    fprintf('   %s.\n',ss);
                    fprintf('In a student list file lines cannot have ');
                    fprintf('length less than %d.',t.STUDENT_ID_LENGTH);
                elseif strcmp(ss(1:7),'-------'),
                    break
                else
                    t.NStud=t.NStud+1;
                    t.StudentList(t.NStud,:)=ss(1:t.STUDENT_ID_LENGTH);
                end
```

```
            end
            fclose(t.sfile);

            % removing empty (zero) lines
            t.StudentList=t.StudentList(1:t.NStud,:);

            % Sort StudentList before starting to do anything with it.
            t.StudentList=mysort(t.StudentList,t.STUDENT_ID_LENGTH-2,t.STUDENT_ID_LENGTH);
            t.StudentList=t.StudentList(:,1:t.STUDENT_ID_LENGTH);
        end % function topic (constructor)
    end

end
```

8. Appendix C: Example of top-file

The following example of top-file shows what kind of information may go into top-file.

```
duedate='Sunday, 9 November 11pm';
rseed=247; % Random seed of the assignment.
PercentMarks=10;
NAltAns=5;
NRepeat=1;
GOOD_ANSWER_ATTEMPTS=80;
%year=2014;
%semester=2;



\documentclass[12pt]{article}
%\usepackage{euscript}
\usepackage{amsmath}
\usepackage{amssymb}
\usepackage{array}
%\usepackage{graphicx}
\input ../../../../Nur/MyTexUtils/Macros.tex
%\newcommand{\makeslide}{}
\textwidth 16cm \oddsidemargin 0cm \textheight 25cm \topmargin
-1.5cm \pagestyle{empty}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\newcounter{nques}
\setcounter{nques}{0}
\newcommand{\problA}[2]{\bigskip \addtocounter{nques}{1} %
   \noindent \mbox{{\bf\arabic{nques}.}\,[$#2$m]}}
\newcommand{\problB}[3]{\bigskip \addtocounter{nques}{1} %
   \noindent \mbox{{\bf\arabic{nques}.}\,[$#2$m,$-#3$m]}}
\renewcommand{\theenumi}{\roman{enumi}}
\renewcommand{\labelenumi}{(\theenumi)}
\newcommand{\noitem}[2]{\ #2}% \ \margcom{\bf (#1)}}
\newcommand{\problemtype}[1]{}  %\margcom{\tiny \sf (#1)}}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

\newcommand{\fl}{{\it fl}}
\renewcommand{\gcd}{\mathrm{GCD}}
\newcommand{\lcm}{\mathrm{LCM}}
\newcommand{\proof}{{\it Proof.} \ }
\renewcommand{\iff}{\Leftrightarrow }
\renewcommand{\id}{\mathrm{id}}
\newcommand{\congn}[2]{#1 \equiv #2 \ \ (\,\mathrm{mod} \ n)}
\newcommand{\congruent}[3]{#1 \equiv #2 \ \ (\,\mathrm{mod} \ #3)}

% Before \begin{document} Spike introduces three
% LaTeX constants: \topiccode, \topicname, \NAss.
% For example:
% \newcommand{\topicname}{Algebra}
% \newcommand{\topiccode}{MATH2712}
% \newcommand{\NAss}{1}

\begin{document}
\bigskip
\newpage
\begin{center} \large Flinders University \Year\ \ Semester \Semester \\
   \topicname \ \topiccode \\
  \bigskip
```

```
  {\bf \bf Assignment \NAss \\ \today}
\end{center}

Please spend a few minutes to read the following text before %
you submit your assignment.
%This may save hours of your time later.

$\bullet$ This assignment constitutes \PercentMarks\% of the total %
topic assessment. Hence, the mark for this assignment will be
accordingly scaled.
%To solve the assignment problems you can use a calculator or MATLAB
%unless it is specified what to use or not to use.

$\bullet$ This assignment is due on \underline{\DueDate.}

$\bullet$ This is a multiple choice assignment and as a result only
answers are marked. The working is not marked.
Therefore, you need to submit only answers to questions, not the working.

$\bullet$ You should submit your answers by email. In the subject of your
email  please give the topic code {\tt \topiccode}, the assignment number
{\tt A\NAss}, your student ID and your answers in
\underline{this order}. The topic code and the assignment number must be
separated by only one space.

For example, if your student ID is 2028022, then the subject line should
look as follows:
\begin{center} \tt
  \topiccode\ A\NAss\ 2028022 ECDCEBADCEADBC...
\end{center}
In this case the answer to the first question is E,
the answer to the second question is C, and so on.
If you do not want to answer a question then put a dot or an X
instead of an answer.

Since one mistake in a string of answers may strongly alter the assignment
mark, it is desirable that the answers from the email's subject are
duplicated in your e-mail. This would allow to easily settle any possible
disputes.

For example, in the case of answer string given above you should
write in the email body a text which may look as follows:

\begin{center} \tt
  Q1  E \\
  Q2  C \\
  Q3  D \\
  ...
\end{center}

Another option is to scan and attach your working to the email.

$\bullet$ What happens to your submission? If you submit answers to an
assignment correctly and in due time then
firstly you will get an automatic notification that your submission has
been received, secondly your submission
gets to a special folder where it will be processed by a computer
program. I don't read your emails and
the computer program which processes assignment submissions also ignores
the email's text. Hence, you don't need
to write in the email something like ``Hi ...'', but you need to duplicate
your answers in the email body
```

```
to resolve possible disputes.

$\bullet$ Your submission will be marked by a marking program. This program
will reject your submission in any of the following cases:
\begin{enumerate}
  \item There is a mistake in the topic code ({\tt\topiccode} for this
assignment) or in the assignment number ({\tt A\NAss} for this assignment)
  \item The topic code and the assignment number are separated by anything
else except one space as shown above;
  \item The student ID from the subject line does not match with student
ID's of any students enrolled in this topic;
  \item If the answer string which follows after the student ID contains
any character except the following ten characters: {\tt ABCDEFGH.X};
  \item The number of answers in the answer line differs from the number
of questions in the assignment.
\end{enumerate}

For example, if you are submitting an assignment \NAss\ of the topic
{\tt\topiccode} and this assignment contains 12 questions and your student
ID is {\tt 2028022} then the following subject lines are correct
in the sense that they are not rejected by the marking program.

\begin{center} \tt
  \topiccode\ A\NAss\ 2028022 AEECDDABCAAA \\
  \topiccode\ A\NAss\ 2028022 AEECD.ABC..A \\
  \topiccode\ A\NAss\ 2028022 AEBCD.ABCXEA \\
\end{center}

The following are a few examples of subject lines which will be rejected
by the marking program with explanation of why they are rejected.
\bigskip

{ \tt
\noindent
\topiccode\ \ A\NAss\ 2028022 AEECDDABCAAA
     \quad (two spaces before the assignment number) \\
\topiccode A\NAss\ 2028022 AEECDDABCAAA
     \quad (no space before the assignment number) \\
\topiccode,A\NAss\ 2028022 AEECDDABCAAA
     \quad (a comma before the assignment number) \\
   MATH2912 A\NAss\ 2028202 AEBCD.ABCXEA    \quad (wrong topic code) \\
\topiccode\ A9 2028022 AEECDDABCAAA    \quad (wrong assignment number) \\
\topiccode\ A\NAss\ 2028202 AEBCD.ABCXEA
     \quad (a typo in student ID) \\
\topiccode\ A\NAss\ 2028022 AEECDDABCCB\ \
      \quad (incorrect number of answers) \\
\topiccode\ A\NAss\ 2028022 AEECDDABCCBEDD
      \quad (incorrect number of answers) \\
\topiccode\ A\NAss\ 2028022 AEECD DABCC BE
      \quad (a space in the answer line) \\
\topiccode\ A\NAss\ 2028022 AEECDDA, Q8[BCC] BE
      \qquad (incorrect characters) \\
   MATH2912 A9 2028202 AEECDDA, Q8[BCC] BE
      \qquad (everything is incorrect:) \\
}

$\bullet$
{\bf Copying answers of someone else or allowing someone else to copy
your answers is a breach of academic
integrity}. Giving hints to correct answers is also prohibited.
{\bf By submitting this assignment you declare that answers you
submit are your own work.}
```

```
$\bullet$
The number in square brackets after a
question number shows the number of points which you get for a
correct answer to this question. If this positive number in the
square brackets is followed by a negative number, then this means
that there is a penalty for a wrong answer to this question, the
penalty being the second negative number; otherwise there is no
penalty for a wrong answer. For example, [2m,$-2$m] means that you
get two points for a correct answer, zero points for no answer and
$-2$ points for an incorrect answer.

$\bullet$ Each student has his/her own version of the assignment
(with randomized numbers). You should find your assignment in this file
using the last three digits of your student ID:
they appear in the right top corner of each page of your assignment.

$\bullet$ If your answer to an assignment question is different from all
the multiple choice answers,
and you still believe that your answer is in fact correct (which is
unlikely but possible), then please put X instead of
a multiple choice answer A,B,... and give your answer in the email.

$\bullet$ If you believe that one (or more) of the questions in your
assignment was marked incorrectly, then you may challenge your mark.
To challenge an assignment mark
send me an email with the following information: topic code, assignment
number, assignment question number, the answer which you gave and the
correct answer which you are challenging. I'll check your challenge and
if it is correct, you will get your deserved mark. If your claim is
incorrect, I'll explain why.

Every student is allowed to make two incorrect challenges (per topic,
per semester). After these two incorrect challenges I reserve the right
not to answer your claims. But unlike the tennis system, if you make a
correct challenge, the number of challenges you have will increase by one.

Correct challenges are very welcome, since they help to improve the system.

$\bullet$
\noindent Consulting times: Tuesday, 2pm--4pm. \\
Venue: Tonsley T1 4.42. \\
My email \ \verb!nurulla.azamov@flinders.edu.au!
```

School of Computer Science, Engineering and Mathematics, Flinders University, Bedford Park, 5042, SA Australia.

E-mail address: nurulla.azamov@flinders.edu.au